

MEC VS MCC: PERFORMANCE ANALYSIS OF REAL-TIME APPLICATIONS

MICAEL DUARTE MARINHO SOARES

julho de 2019

MEC vs MCC: PERFORMANCE ANALYSIS OF REAL-TIME APPLICATIONS

Micael Duarte Marinho Soares



Departamento de Engenharia Electrotécnica

Mestrado em Engenharia Electrotécnica e de Computadores

Área de Especialização em Telecomunicações

2019

Relatório elaborado para satisfação parcial dos requisitos da Unidade Curricular de
Tese/Dissertação do Mestrado em Engenharia Electrotécnica e de Computadores

Candidato: Micael Soares, Nº 1000167, 1000167@isep.ipp.pt

Orientação científica: Prof. Doutor Jorge Mamede, jbm@isep.ipp.pt



Departamento de Engenharia Electrotécnica
Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização em Telecomunicações

2019

To my wife Ana and my children Diogo and Leonor

Agradecimentos

Foremost, I would like to express my sincere gratitude to my advisor Prof. Jorge Mamede for the continuous support of my MSc study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

I would also like to thank Bruno Santos for his collaboration in the developments carried out during the first part of this thesis. Without his passionate participation and input, this work could not have been successfully conducted.

I would also like to acknowledge the OpenEDGEComputing team, I am gratefully indebted for their valuable comments on this thesis.

Finally, I must express my very profound gratitude to my parents, my sister, my wife and my children for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Resumo

Hoje em dia, numerosas são as aplicações que apresentam um uso intensivo de recursos empurrando os requisitos computacionais e a demanda de energia dos dispositivos para além das suas capacidades. Atentando na arquitetura Mobile Cloud, que disponibiliza plataformas funcionais e aplicações emergentes (como Realidade Aumentada (AR), Realidade Virtual (VR), jogos online em tempo real, etc.), são evidentes estes desafios directamente relacionados com a latência, consumo de energia, e requisitos de privacidade.

O Mobile Edge Computing (MEC) é uma tecnologia recente que aborda os obstáculos de desempenho enfrentados pela Mobile Cloud Computing (MCC), procurando solucioná-los. O MEC aproxima as funcionalidades de computação e de armazenamento da periferia da rede.

Neste trabalho descreve-se a arquitetura MEC assim como os principais tipos soluções para a sua implementação. Apresenta-se a arquitetura de referência da tecnologia *cloudlet* e uma comparação com o modelo de arquitetura ainda em desenvolvimento e padronização pelo ETSI.

Um dos propósitos do MEC é permitir remover dos dispositivos tarefas intensivas das aplicações para melhorar a computação, a capacidade de resposta e a duração da bateria dos dispositivos móveis. O objetivo deste trabalho é estudar, comparar e avaliar o desempenho das arquiteturas MEC e MCC para o provisionamento de tarefas intensivas de aplicações com uso intenso de computação. Os cenários de teste foram configurados utilizando esse tipo de aplicações em ambas as implementações de MEC e MCC. Os resultados do teste deste estudo permitem constatar que o MEC apresenta melhor desempenho do que o MCC relativamente à latência e à qualidade de experiência do utilizador. Além disso, os resultados dos testes permitem quantificar o benefício efetivo tecnologia MEC.

Palavras-Chave

Mobile computing, Cloud Computing, Edge Computing, Cloudlets, Mobile Edge Computing, máquinas virtuais (VM), provisionamento de VM, tempo de resposta, tramas por segundo (FPS).

Abstract

Numerous applications, such as Augmented Reality (AR), Virtual Reality (VR), real-time online gaming are resource-intensive applications and consequently, are pushing the computational requirements and energy demands of the mobile devices beyond their capabilities. Despite the fact that mobile cloud architecture has practical and functional platforms, these new emerging applications present several challenges regarding latency, energy consumption, context awareness, and privacy enhancement.

Mobile Edge Computing (MEC) is a new resourceful and intermediary technology, that addresses the performance hurdles faced by Mobile Cloud Computing (MCC), and brings computing and storage closer to the network edge.

This work introduces the MEC architecture and some of edge computing implementations. It presents the reference architecture of the cloudlet technology and provides a comparison with the architecture model that is under standardization by ETSI.

MEC can offload intensive tasks from applications to enhance computation, responsiveness and battery life of the mobile devices. The objective of this work is to study and evaluate the performance of MEC and MCC architectures for provisioning offload intensive tasks from compute-intensive applications. Test scenarios were set up with use cases with this kind of applications for both MEC and MCC implementations. The test results of this study enable to support evidence that the MEC presents better performance than cloud computing regarding latency and user quality of experience. Moreover, the results of the tests enable to quantify the effective benefit of the MEC approach.

Keywords

Mobile computing, Cloud Computing, Edge Computing, Cloudlets, Mobile Edge Computing, Virtual Machines, offloading, VM provisioning, VM instances, VM overlay, response time, FPS.

Contents

AGRADECIMENTOS.....	I
RESUMO.....	II
ABSTRACT.....	III
CONTENTS.....	IV
LIST OF FIGURES	VII
LIST OF TABLES	X
ACRONYMS	XI
1. INTRODUCTION	1
2. THEORETICAL BACKGROUND	3
2.1.MARKET DRIVERS.....	4
2.2.TOWARDS 5G	9
2.3.MOBILE COMPUTING.....	11
2.4.FRAMEWORKS DEPLOYMENTS	19
2.5.SUMMARY	21
3. THE MEC AND CLOUDLETS	23
3.1.STANDARDIZATION	23
3.2.CLOUDLET.....	29
3.3.PROBLEM STATEMENT	31
3.4.SUMMARY	32
4. CLOUDLET IMPLEMENTATION.....	33
4.1.CLOUDLET ARCHITECTURE.....	34
4.2.OPENSTACK	37
4.3.OPENSTACK EXTENSION - OPENSTACK ++.....	40
5. PERFORMANCE TESTS SETUP	49
5.1.CLOUD PLATFORM.....	50
5.2.TESTBENCH SCENARIO.....	52
5.3.USE CASE 1 - FLUID MOBILE APPLICATION.....	53
5.4.USE CASE 2 – FACE\$WAP MOBILE APPLICATION	58

6. ANALYSIS OF THE RESULTS	69
6.1.TEST RESULT USE CASE - FLUID	70
6.2.TEST RESULT FROM USE CASE - FACE SWAP	77
7. CONCLUSION	87
REFERENCES	89
HISTORY	101

List of Figures

Figure 1 - IoT architecture [5]	5
Figure 2 - Augmented Reality [5]	6
Figure 3 - Active device location tracking [5]	7
Figure 4 - Intelligent video analytics [5]	7
Figure 5 - RAN-aware content optimization [5]	8
Figure 6 - 5G requirements	9
Figure 7 - Uses cases vs speed and response time [9]	10
Figure 8 - Architecture of Mobile Edge Networks [32]	14
Figure 9 - Fog computing architecture	16
Figure 10 - Cloudlet Architecture	17
Figure 11 - Mobile Edge Computing architecture	18
Figure 12 - MEC framework	27
Figure 13 - MEC reference architecture [65]	28
Figure 14 - Cloudlet architecture	30
Figure 15 - Cloudlet framework proposed	30
Figure 16 - Cloudlet vs ETSI MEC reference architecture	31
Figure 17 - Three-tier architecture for code offload	33
Figure 18 - VM-based cloudlet architecture	34

Figure 19 - VM overlay creation	35
Figure 20 - OpenStack software diagram [85]	37
Figure 21 - Conceptual OpenStack architecture	39
Figure 22 - Openstack++ cloudlet platform	40
Figure 23 - Cloudlet API call hierarchy [85]	41
Figure 24 - OpenStack++ final configuration setup	42
Figure 25 - OpenStack Dashboard	43
Figure 26 - Import Base VM Image Process	44
Figure 27 - Base VM files metadata	45
Figure 28 - Resume Base VM creation process	46
Figure 29 - Overlay creation process	47
Figure 30 - Cloudlet VM Synthesis creation process	48
Figure 31 - VM Instance Handoff setup creation	48
Figure 32 - Performance evaluation diagram	49
Figure 33 - AWS EC2 Dashboard	51
Figure 34 - Evaluation infrastructure setup	52
Figure 35 - Fluid client application	54
Figure 36 - Fluid client application running	55
Figure 37 - Communication sniffing	56
Figure 38 - VM Synthesis process	57
Figure 39 - FaceSwap android application	59

Figure 40 - Server configuration	59
Figure 41 - FaceSwap Training session	60
Figure 42 - FaceSwap choose option menu	61
Figure 43 - Swap person selection	62
Figure 44 - Application client server process	63
Figure 45 - Server launching control	64
Figure 46 - Cognitive engine communications	65
Figure 47 - FaceSwap image metadata	66
Figure 48 - VM overlay metadata files	72
Figure 49 - CDF for response time and frame rate for Fluid - Cloudlet	73
Figure 50 - CDF for response time and frame rate for Fluid - Cloud	75
Figure 51 - Cloud and Cloudlet test result comparison for Fluid	76
Figure 52 - CDF for response time and Frame rate for FaceSwap - Cloudlets	80
Figure 53 - CDF for response time and frame rate for FaceSwap - Clouds	82
Figure 54 - Cloud and Cloudlet test result comparison for FaceSwap	84
Figure 55 - CPU usage while FaceSwap running	85

List of Tables

Table 1 - Comparison of MEC and MCC systems [23]	13
Table 2 - Comparison of Cloudlets, Fog and MEC approaches	15
Table 3 - Smartphone specification	52
Table 4 - Cloudlet specifications	53
Table 5 - Servers instance configuration type	70
Table 6 - Cloudlet test results for Fluid	74
Table 7 - Cloud test results for Fluid	74
Table 8 - Instance type configuration	78
Table 9 - Cloudlet test results for FaceSwap	78
Table 10 - Cloud test results for FaceSwap	80

Acronyms

3GPP	–	3rd Generation Partnership Project
5G	–	Fifth Generation Networks
AD	–	Autonomous Driving
AMI	–	Amazon Machine Image
AP	–	Access Point
API	–	Application Programming Interface
AR	–	Augmented Reality
AWS	–	Amazon Webservices Services
BS	–	Base Station
CACTSE	–	Cloudlet Aided Cooperative Terminals Service Environment
CPU	–	Communications Processor Unit
DNN	–	Deep Neural Network
DNS	–	Domain Name System
EC2	–	Amazon Elastic Compute Cloud
ETSI	–	European Telecommunications Standards Institute
IaaS	–	Infrastructure-as-a-Service
ICT	–	Information and Communications Technology
IoT	–	Internet of Things

ISG	– Industry Specification Group
KVM	– Kernel-based Virtual Machine
LCM	– Lifecycle Management
LTE	– Long Term Evolution
MCC	– Mobile Cloud Computing
MEC	– Mobile Edge Computing
MOCHA	– Mobile Cloud Hybrid Architecture
NFV	– Network Functions Virtualization
OEC	– Open Edge Computing
OS	– Operative System
PaaS	– Platform as a Service
PCA	– Principal Component Analysis
PoC	– Proof of Concept
QoE	– Quality of Experience
QoS	– Quality of Service
RAN	– Radio Access Network
RTT	– Round Trip Time
RNC	– Radio Network Controller
SDN	– Software Defined Networking
SDO	– Standards Developing Organizations

SM	–	Service Manager
SPH	–	Smoothed Particle Hydrodynamics
UE	–	User Equipment
TCP	–	Transmission Control Protocol
UHD	–	Ultra High Definition
VIM	–	Virtualization Infrastructure Manager
V2I	–	Vehicle-to-Infrastructure
V2V	–	Vehicle-to-Vehicle
V2X	–	Vehicle-to-Everything
VIM	–	Virtualization Infrastructure Manager
VM	–	Virtual Machine
VR	–	Virtual Reality
WAN	–	Wide Area Network
WAP	–	Wireless Access Point

1. INTRODUCTION

1.1 MOTIVATION

The development of telecommunication networks has led to the emergence of new applications on mobile devices. Some applications are resource-intensive and, consequently, push the computational requirements and energy demands of mobile devices beyond their capabilities.

It is in this context that Mobile Cloud Computing (MCC) arises as a practical solution for offloading mobile devices. Despite the fact that mobile cloud-based architectures provide functional platforms, those applications present several challenges regarding latency, energy consumption, context awareness, and privacy. Mobile Edge Computing (MEC) is a new resourceful and intermediary technology, that addresses the performance hurdles faced by MCC and brings the computing and storage resources closer to the network edge.

1.2 OBJECTIVE

The objective of this work is to study and evaluate the performance of MEC and MCC architectures for provisioning offload intensive tasks from compute-intensive applications. Test scenarios shall be defined and use cases executed with that kind of applications for both

MEC and MCC implementations, in order to quantify the performance achievements of each approach.

1.3 ORGANIZATION

This thesis reports a study carried out to evaluate the performance of resource-demanding applications in an edge and cloud-based approaches. This work is organized as follows:

- In Chapter 2, an overview of the actual scenario exposes the problem background in order to motivate the necessity of Mobile Edge Computing. It presents also the markets drivers which will benefit from the use of this technology, and also make an approach to the emerging 5G technology where this new architecture will be crucial to achieving the respective objectives. A summary presents the difference between the cloud and the edge computing, and an analysis related to the key features of the 3 principal frameworks of the edge computing: fog computing, cloudlets, and mobile edge computing.
- In Chapter 3, an introduction summary addresses the standard efforts made by the community to orchestrate and normalize a reference architecture and framework. Secondly, this thesis presents the cloudlet architecture, compares it with the reference architecture proposed by the European Telecommunications Standards Institute (ETSI). In conclusion of the previous elements, we present the Thesis Statement.
- Chapter 4 addresses the technical challenges of cloudlets by offloading computation intensive part of the application and describes all processes executed by the cloudlets.
- In Chapter 5, we present the testbench scenario used to study and evaluate the performance of MEC and cloud solutions over 2 uses cases. An analysis is made of two use cases using compute intensive task applications. An explanation is provided for each application regarding the client side and the server side with the offload intensive task part.
- In Chapter 6, this thesis presents the result tests for each use cases scenarios.
- Chapter 7 concludes the dissertation through the analysis of the values obtained and explains future work.

2. THEORETICAL BACKGROUND

In the last decade, Cloud Computing has emerged as a new paradigm in computer science delivering centralized services to end users. Cloud computing provides a shared pool of resources available all time, as centralized computing, storage and network management in the Cloud, Data Centers, backbone IP networks or cellular core networks [1, 2].

All kind of smart devices and sensors technology are connected, and this exponential growth result on a challenge for Cloud computing in order to meet many new requirements in the emerging Internet of Things (IoT).

IoT is generating a huge quantity of data that needs to be analyzed, processed, transformed, stored and answered on an unprecedented scale and in a short time. Today clouds have become an indispensable part of that process; however, clouds centrally deployed but providing services on a global scale need to process an enormous amount of data. In addition, the infrastructure uses an end-to-end topology, so all these processes are supported from the Cloud and Data Centers to the IoT devices and end users.

As the physical distance between the Users and the Cloud increases, transmission latency increases with it, increasing response time and stressing out the user. On top of that, the processing rate in this environment considerably depends on the performance of the equipment.

By 2020, an estimation states that 50 billion of smart devices connected will exist and the volume of data will grow exponentially [3]. Traffic of 1,5 GB of data is expected per person per day [4]. All this explosion of data cannot be send to the Cloud, because it is not affordable to transport all this data in that timeframe. We are now at a transition point to drive an architecture change, where all major contributors are working together in order to implement new technologies.

Upon now, Mobile Cloud Computing (MCC) is mainly focused around the devices generating data and its transmission to the cloud for storage and then compute cycles are used to extract its value. However, the connection to a cloud does not present affordable latencies for many type of applications, when settled thousands of kilometers from the user.

In this context, a distributed solution based on local servers placed at the network edges, providing computation power, analytics and storage capacity so that the mobile devices have a minimum computation effort and lower latency. Considering crucial for servers to be located close to the user, Mobile Edge Computing (MEC) [4] has emerged as a fundamental technology that will permit to develop the 5G vision and extract better benefits from the Internet of Things, Tactile Internet or Internet of Me [6]. Community and researchers from Industry and Academy, are working together to implement, test, promote, and normalize MEC technology.

2.1. MARKET DRIVERS

This Thesis started when researchers and the European Telecommunications Standards Institute (ETSI) were progressing in this young field and journey. It aims to present the benefits of Edge Computing in mobile networks and a few use case of applications that are deployable in the closer term. It also provides an overview of the MEC architecture, its deployment options, and presents results on the deployment such applications through MEC technology.

The initial objective of Mobile Edge Computing is to provide Cloud Computing and IT services to the mobile environment anywhere and anytime, with data stored outside the mobile devices [26].

Network operators and content providers can provide and exploit services to integrate across MEC platform. The main goal is to achieve a better user experience interaction and response by accessing faster applications through a nearby position. Moreover, information and services can be deliver directly and do not need to rely exclusively on cloud services anymore.

The key players in MEC are infrastructure and device manufacturers, software providers, applications developers, Telecoms and Network Operators [31].

To networks operators, which are always searching for new revenues, MEC serves an important role in improving wireless system performance and reducing the cost of operation. They are already in the transition process through key technologies: Software Defined Networking (SDN) and Network Functions Virtualization (NFV) [97]. For hardware and software developers, MEC can bring new opportunities to create applications and consumer products through to promote mobile edge platforms and virtualization infrastructure and become innovation leaders in an otherwise fully commoditized market.

Since all community is working together in MEC definition, the standardization is moving forward rapidly in last months. In [55], the authors present the main challenges for MEC scenarios, such as data interoperability, resource management, orchestration, service discovery and security.

The following section presents some use cases and scenarios that illustrate the performance improvements provided by the utilization of MEC technology.

2.1.1. USE CASES: USER SIDE

IoT intends to connect smart devices to the Internet in order to exchange information and data, such as identification, location, monitoring and management [56].

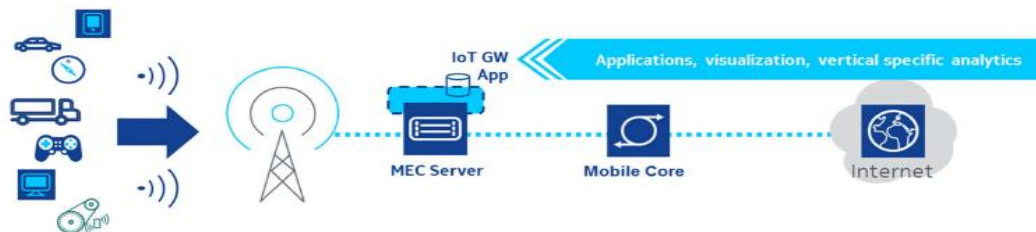


Figure 1 - IoT architecture [5]

IoT is a network that interconnects physical devices, sensors, actuators or electronics with embedded software which can exchange data through wireless communication. Consequently, it provides connections and networkings to vehicles, transport services, community services or societies infrastructure [57].

There is a need to improve latency, response time and battery life time, due to various protocols used, the amount of message sent and the analytics parts regarding the data collected. MEC server provides the capability to solve these challenges.

By providing efficient delivery of local content, new types of applications can be promoted, such as gaming, virtual reality and augmented reality. Indeed, the rendering could be implemented on the mobile phone itself, but the heavy computation can overwhelm the limited processing capability of the phone. Besides, offloading physical simulation and artificial intelligence to a remote cloud server might incur high latency time. MEC can provide both computation power and proximity. Augmented Reality (AR) is an example that merges the real world's view with some computed generated sensors, such as data, video, sound and graphics [5]. It allows to interact dynamically with the user, since the user is able to view the real world and to digitally manipulate some virtual objects. In order to overlay information from the phone camera, localized content has to be rendered quickly. The processing can be performed on the MEC server as a requirement in order to improve high speed and low latency.

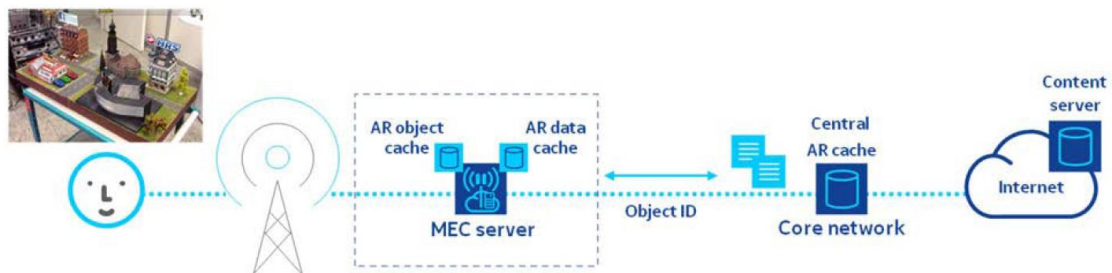


Figure 2 - Augmented Reality [5]

Connected Vehicles is a key trend market that will grow in the next years, through the support to Vehicle-to-everything (V2X) communication. All kind of information that affect vehicles can be collected, such as road conditions, route prediction, collision warnings. As the number of connected vehicles increases and the technology evolves, the volume of data will continue to increase also, so the necessity to reduce latency and maximize Quality of Experience (QoE).

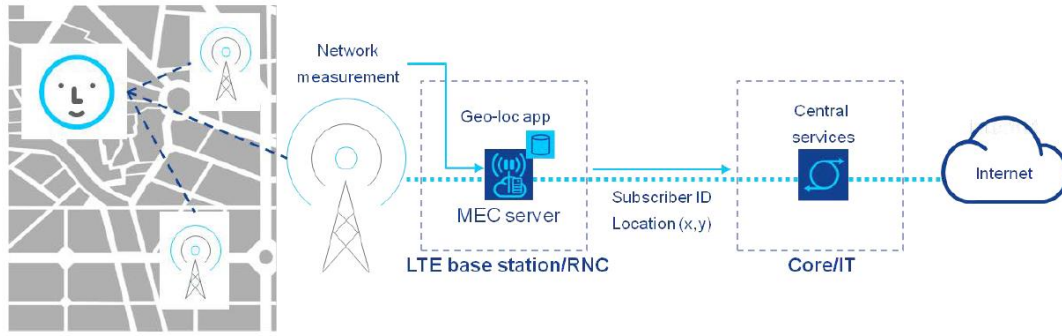


Figure 3 - Active device location tracking [5]

In that context, utilization of MEC technology can push applications, data and services closer to vehicles and will result in the applications acceleration over the vehicles [55]. The MEC application can operate as a highly distributed roadside unit that support drivers with real-time useful information and the safety improvement of the roads, as presented in Figure 3.

2.1.2. USE CASES: PROVIDERS SIDE

In the same geographical area, many users tend to consume the same content at the same time, such as shared larges files, high-definition videos. Therefore, all these contents can be cached locally at the edge hosts to reduce drastically the backhaul network capacity. Furthermore, Quality of Service (QoS) and Quality of Experience (QoE) can be improved proactively by moving cached data to mobile edge hosts in anticipation of user movement.

In the next years, the massive influx of IoT devices may overwhelm backhaul network as all amount data and services collected by sensors and mobile devices are sent to the remote Cloud servers.

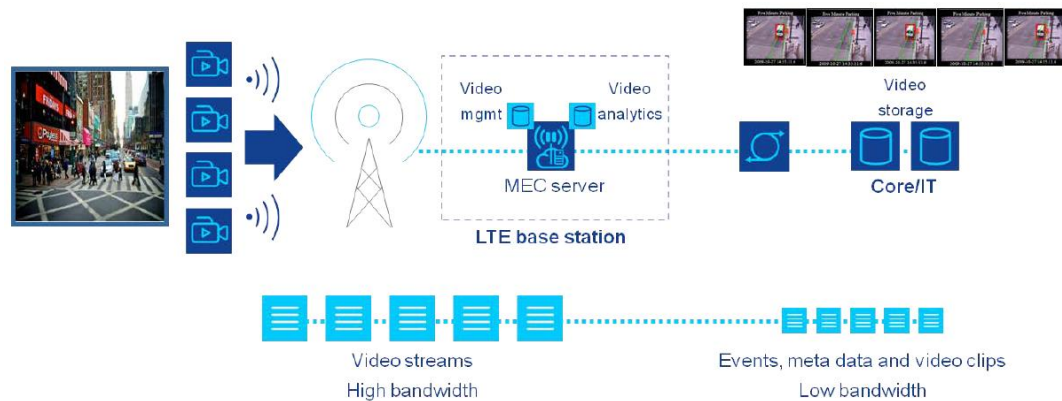


Figure 4 - Intelligent video analytics [5]

Instead of that, mobile edge hosts can process a first data aggregation and analytics and forward only necessary information to backend servers, such example is shown in Figure 4.

Another use cases concerns the mobile media streaming with bandwidth feedback, information provides throughput guidance to a video server. As the available capacity can vary instantly and consistently in a mobile network, video quality of experience of user is not optimal. Indeed, Transmission Control Protocol (TCP) is not fast enough to detect quick variations, drastic fluctuations leading to an underutilization of the radio resources. MEC technology can inform video server of the optimal server to use the radio conditions for a particular video stream or user.

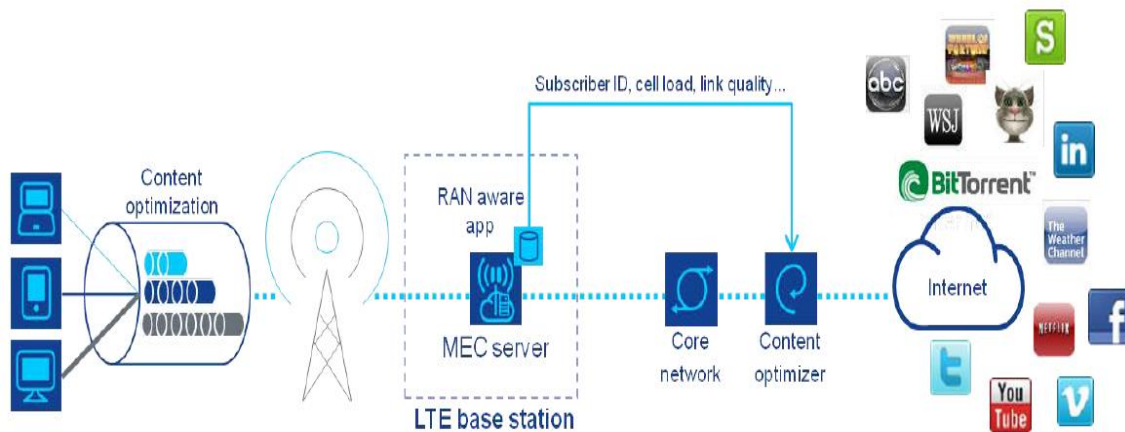


Figure 5 - RAN-aware content optimization [5]

MEC server example presented in Figure 5 improves mobile backhaul optimization since it can determinate or estimate the throughput, traffic and performance at the real-time Radio Access Network (RAN) level and then be made available to the backhaul network.

Thus, the backhaul can be optimized through techniques such as application traffic shaping, traffic routing, and capacity provisioning.

MEC provides more effective location-based services in two ways. First, from the received signal strength and analytical techniques it allows user location tracking. Second, applications can use the user location and behavior pattern, to give recommendations. It may also utilize advanced machine-learning techniques and interface with big data analysis at backend servers to further improve the accuracy and usefulness of its recommendations.

2.2. TOWARDS 5G

In the past decade, Wireless communications and networking evolved significantly driven by the huge growth of mobile devices and mobile traffic. Remote data centers were allowed to run computing services for mobile devices since wireless communication presented a high bitrate and reliability. It resulted in the research area called Mobile Cloud Computing (MCC).

However, there are known limitations of MCC, as latency resulting from the distance between the end user and the remote cloud data center. New mobile application and devices are emerging that are latency-critical, so they will not be suitable for the MCC. In the last years, new technologies move cloud computing closer to the user at the Edge.

The fifth generation networks (5G) is currently under development and will hit the market at the horizon of 2020. Figure 6 presents the target of 5G, which is to reach high speed (1 Gbps), low power and latency (1 ms or less) for massive IoT, tactile internet and robotics. Computational capacity will be deployed in many kind of edge devices, like wireless access points (WAPs), Base Stations (BSs) or even smartphones, tablets or laptops. All these devices will be using computation and storage resources available at network edges, which allow a permanent mobile computing.

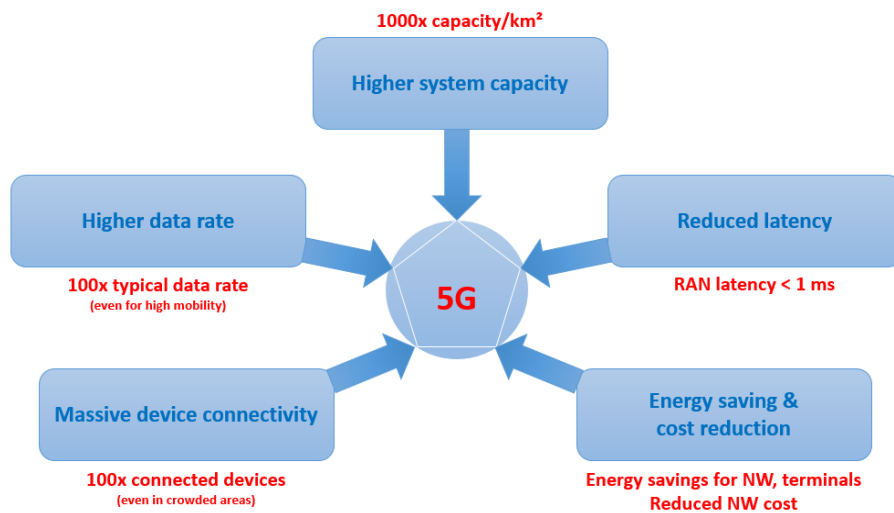


Figure 6 - 5G requirements

In 5G, systems will need to meet requirement to fully support the 4C – communications, computing, control and content delivery. 5G networks expand broadband wireless services beyond the advent and evolution of mobile internet, Information and Communications Technology (ICT), IoT and critical communications segments. New emerging application and services for 5G require specific and challenging high access speed and low latency, such as Autonomous Driving (AD), Augmented Reality (AR), Virtual Reality (VR), Tactile Internet, real-time online gaming and Ultra High Definition (UHD) video streaming shown in Fig. 7 [9].

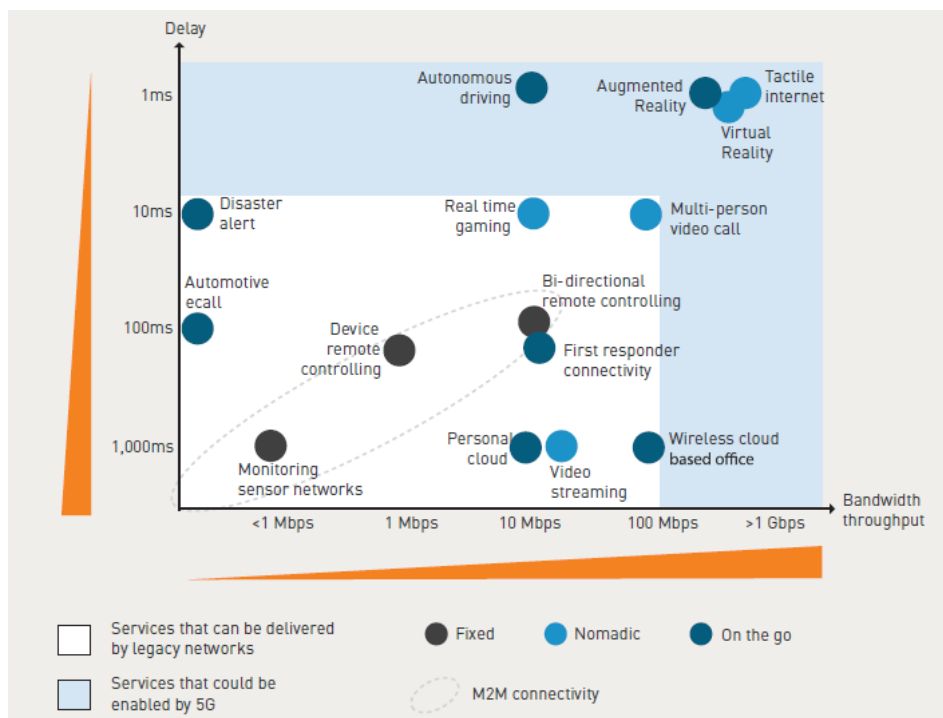


Figure 7 - Uses cases vs speed and response time [9]

Smart mobile devices have limited resources for computing, communication and storage, and have to rely on Clouds or edge devices for enhancing their capabilities. All community have agreed that 5G requirements for a few milliseconds for computing and communication makes cloud computing inadequate. The enormous quantity of data exchange between end users and remote clouds will saturate backhaul networks, and the solution is to bring down all these computation, communication and traffic to the network edges. The explosion of applications for IoT, social networks and content delivery turn on the necessity that information generated locally needs to be consumed locally as a key factor for next generation network concept [10].

2.3. MOBILE COMPUTING

In the past two decades, mobile cellular networks have experienced four generations of evolution following the advent of the ICT and telecommunications technology. At the same time, mobile devices are also constantly evolving but it remains that the computational power, limited storage and low battery life are limitations that are especially critical for resources demanding applications.

As smart devices and applications has emerged, new requirements have appeared to fulfil end user quality of service and experience.

In the future 5G system, as the traditional Base Station (BS) cannot fulfil these requirements, the mobile network architecture is evolving from BS to device and content network [8]. In this section, we will explain the concept of mobile edge networks, the solutions proposed and finally the advantages.

2.3.1. MOBILE CLOUD COMPUTING

Initially, the main concept of MCC was to provide a centralized computing, storage and network management in the Cloud due to the limited resources available in the mobile devices. So, remote servers executed intensive computation tasks or storage. MCC provided many solutions like mobile learning, healthcare, searching services [11]. Nowadays, MCC continue to offer relevant and resilient services where key characteristics have no substantial impact in the user experience, such as mobile devices energy consumption, network bandwidth, latency, context and location awareness. The Table 1 presents significant differences through the comparison of some key features, as some examples are described below.

Latency: Some parameters are crucial to provide low latency, such as distance propagation, computation resources and bandwidth. Mobile Computing requires transmission between end users and the Cloud, and the distance to go through to remote server can be thousands of kilometers through different kind of technology from mobile network, to backhaul network and internet. MCC has the advantage of offering a higher computational resource, but more users share it.

MCC presents total latency between 30 and 100 ms [12], which is unacceptable for applications, such as autonomous vehicles or real-time online gaming. These applications need latency in order of 1 ms [13]. However, MEC has the potential to reach that time and become the key technology for 5G applications. Compared to MCC, MEC locates at the edge and benefits with low latency and communication free.

Energy Consumption: IoT devices have limited resources and also limited energy storage due to the compact design. Nevertheless, low resources tasks are performed by IoT devices in the main areas of surveillance, health monitoring or crowd-sensing [14]. The key disadvantage of the IoT devices is the frequency to recharge or replace battery. MEC technology is the solution that enables computation offloading at the edge, resulting in an improvement of battery life of the IoT devices.

By offloading computation significant energy saving can be done. In [15], the authors refer to the application *eyeDentify* running over a MEC architecture, that increases up to 44 times the computation capacity. In [16], MEC AR applications achieve a 30-50% increase of battery life.

Context Awareness: Key factor in MEC technology due to the end users are near from edge devices, this provide real-time information regarding location, environment and behavior, such information can be deployed into services to end users [17] [18]. A perfect example for AR application is the Museum Video Guide [19], that provide location awareness and information regarding the piece of art or antiques artefact. Another application provide traffic monitoring, navigation and routing of large number of persons through fingerprints [20].

Privacy enhancement: MEC technology enhance privacy and security for mobile capacity. In MCC systems, the Cloud platforms are large public data centers, such as Amazon EC2 or Microsoft Azure, that have a huge quantity of users information resources provoking possible attacks. In addition, there is a possibility of data leakage and data loss as ownership and data management are separated [21]. As MEC server will not have much information, reducing possible attacks. MEC could act as a small cloud near to the users, resolving sensitive data communication between end users and servers. An example of that is the case of a system administrator that sends critical information to remote data centers [22].

2.3.2. EDGE COMPUTING FRAMEWORKS

The core objective of mobile edge networks is to move resources closer to the network edges. The network resources are computing, storage and caching [40].

MEC is an evolution of MCC and performs computing-intensive tasks and storing massive amounts of data at the edge of the networks.

Table 1 - Comparison of MEC and MCC systems [23]

	MEC	MCC
Server hardware	Small data centers with moderate resources [5], [24]	Large-scale data centers (each contains a large number of highly-capable servers) [25], [26]
Server location	Co-locate with wireless gateways, WiFi routers, and LTE BSs [5]	Installed in dedicated buildings, with size of several football fields [27], [28]
Deployment	Densely deployed by telecom operators, MEC vendors, enterprises, and home users. Require lightweight configuration and planning [5]	Deployed by IT companies, e.g., Google and Amazon, at a few locations over the world. Require sophisticated configuration and planning [25]
Distance to end users	Small (tens to hundreds of meters) [29]	Large (may across the country border) [30]
Backhaul usage	Use not frequent Alleviate congestion [31]	Frequent use Likely to cause congestion [31]
System management	Hierarchical control (centralized/distributed) [32]	Centralized control [32]
Support latency	Less than tens of milliseconds [29], [33]	Larger than 100 milliseconds [34], [35]
Applications	Latency-critical and computation-intensive applications, e.g., AR, automatic driving, and interactive online gaming [5], [35].	Latency-tolerant and computation-intensive applications, e.g., online social networking, and mobile commerce/health/learning [36]–[39].

In MEC architecture, data processing and data storage happen outside of mobile devices [41]. However, new emerging applications represent a serious challenge to MCC in terms of latencies, video download, traffic congestion and capacity that frustrates end users. Businesses need competitive, scalable and secure solutions.

The basic idea is to perform computations and running applications near the mobile user, it reduces network congestion and gets a better performance out of mobile applications. Mobile management will reinforce the reduction of costs and presents new functionalities in the service area, such as controlling enterprise through mobile devices, promoting security and enforcement to the Police Department or the Municipalities maintenance teams.

Regarding congestion, IoT applications and services at the edge enable proximity, providing ultra-low latency, higher bandwidth, real-time access to RAN information and location awareness. Some of these challenges are listed in [41].

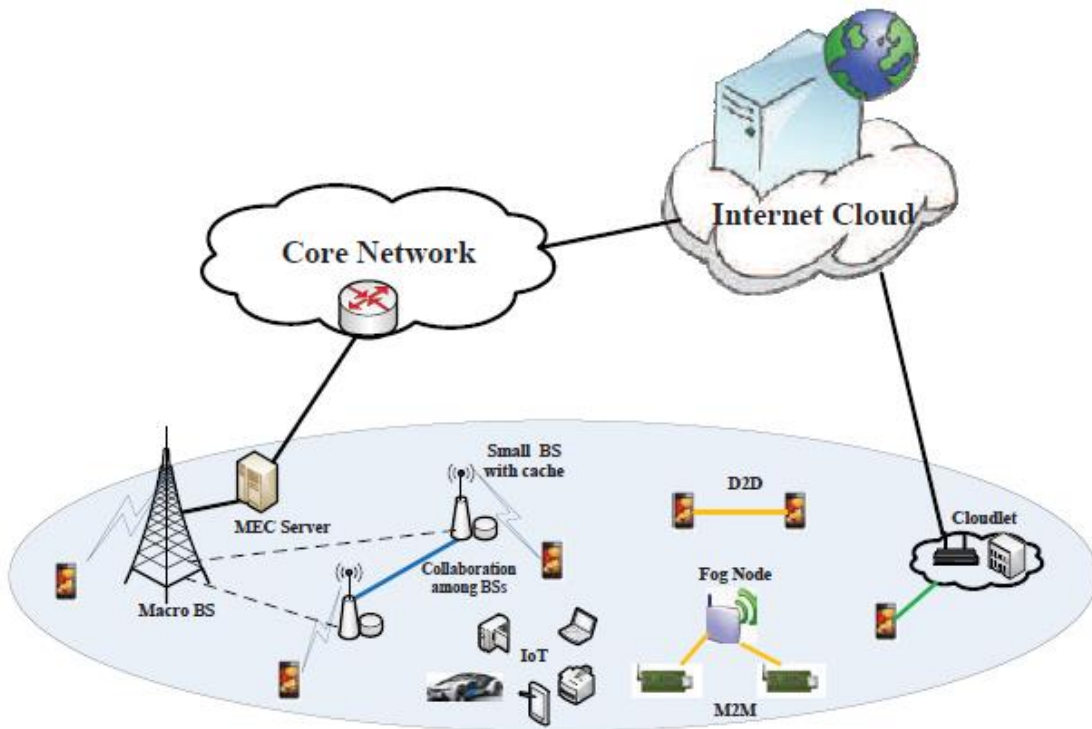


Figure 8 - Architecture of Mobile Edge Networks [32]

As the computing part, Edge computing allows computing capabilities at the network edge, and also efficient and dynamic offloading, data access and context awareness. The community composed by researchers from Industry and Academy have proposed three

different schemes of Edge computing: Fog Computing nodes, Mobile Edge Computing servers and Cloudlets, as shown on Figure 8.

The Table 2 summarizes the main features of these three edge computing technologies, which present some similarities between them.

Table 2 - Comparison of Cloudlets, Fog and MEC approaches

Properties	Cloudlets based approach	Fog Computing approach	MEC approach
Reduce Latency	Y	Y	Y
Reduce Jitter	Y	Y	Y
Multi-Tenancy	Y	Y	Y
With Virtual IaaS Platform	Y	Y	Y
Location	Y	Y	Y
Geographical Distributed	Y	Y	Y
Mobility Support	Y	Y	Y
Inspired from	Tactile Internet	IoT	Mobile World
Extended from Cloud	Y	Y	May or may not
Mostly used with wireless access	May or may not	Y	Y
Focus on-line analytics	May or may not	N	Y
Located between DC and device	Y but can directly run on a device	Y	Y
Improve user experience	Y	Y	Y
N-tier	N = 3	N = 3 or more	N = 2 or 3
Y = Yes, N = No			

FOG COMPUTING

The concept of Fog Computing was introduced by Cisco in 2012, and initially it was considered as an “*extension of the cloud paradigm that provides computation, storage, and networking services between end devices and traditional cloud servers*” [42].

The Open Fog Consortium has made an effort to define a distributed three-tier architecture (end users, fog nodes and centralized Clouds) where each element communicate and interact with each other, as shown in Figure 9. The main objective is to put data close to the end user [43], which reduces latency, improves QoS [44] and provides support for localization, context awareness and mobility support [45].

Fog enables the harvesting of local information analysis and the Cloud performs the coordination and global analytics in order to meet the demands from different segments of

business: consumer, wearable, industrial, enterprise, automobile, healthcare, building, energy. As the fog network architecture is heterogeneous, services can be deployed in various locations at the network edges at a high speed data-rate and through different wireless access technologies [46].

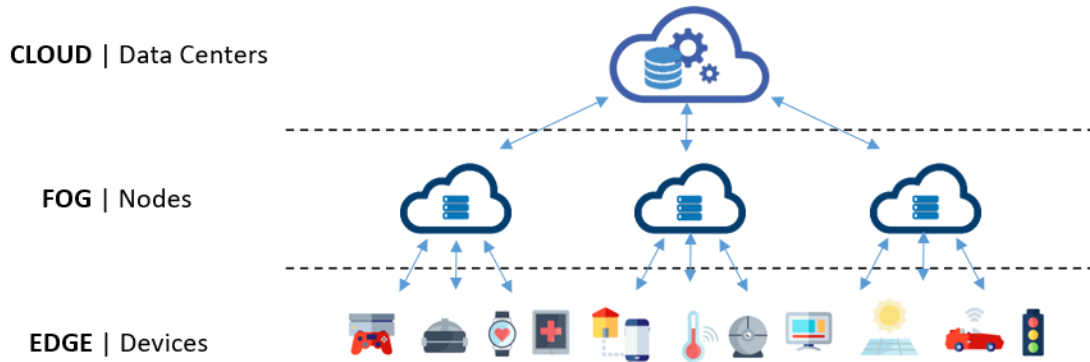


Figure 9 - Fog computing architecture

Originally, Fog Computing was designed to create new applications and services in the context of Internet of Things, such as Big Data analytics systems or smart infrastructure management system [42][47]. Recently, there is a need to extend this concept to other types of services through several studies: augmented reality and real-time video analytics [48], content delivery and caching approaches through Fog computing [49], vehicle systems such as vehicle-to-vehicle (V2V) and Vehicle-to-infrastructure (V2I) and Vehicle-to-everything (V2X) [50] or low-latency augmented interface devices [51].

CLOUDLETS

The concept of Cloudlet was developed by an academic team at Carnegie Mellon University [12], where a prototype was developed as part of a research project called Elijah [52]. The key features of cloudlets are the real-time provisioning of applications to edge nodes through Wi-Fi or cellular networks. It provides also the possibility to smart devices to move through different edge servers and continue to use applications by using handoff of virtual machine images among edge nodes [53].

The main motivation of this solution was to handle the problem that surge from the resource constraint on the mobile devices. The Cloudlets are designed to support applications for

mobile devices through the offloading of the resources due to aggressive tasks and interactions, such as Augmented Reality applications, Cloud games, and Wearable cognitive assistance system like Google Glass, Apple Siri or Google Now. As this solution performs high resource intensive task and faster execution and reduces the communication latency, it is considered as a key solution regarding the emerging of the Mobile Edge Computing architecture and implementation.

The topology design is a third tier architecture composed by mobile devices, Cloudlets and the Cloud, as shown in Figure 11.

In this hierarchy, the Cloudlets are the middle tier and can be considered as a local data centre used to enable localized cloud services, offer high performance and faster access to cloud resources to multiple users simultaneously. Moreover, it provides benefits with high utilization cost issues, large Wide Area Network (WAN) latency and less bandwidth [54].

Open Edge Computing (OEC) was formed as new emerging open source banner from the Carnegie Mellon University and industrial key players, such as Nokia, Intel and Vodafone. This initiative promotes the use of Cloudlets as an enabling technology through the extension of open source codes APIs of the OpenStack platform [55]. This initiative aims to synchronize efforts in Standardize schemes.



Figure 10 - Cloudlet Architecture

MOBILE EDGE COMPUTING

ETSI [22] introduced the concept of Mobile Edge Computing in 2014, which main goal was to standardize a MEC architecture and Application Programming Interfaces (APIs) for 3rd party applications for all major actors of the community [40].

The MEC architecture is based on a virtualized platform that enables application running on top of MEC servers, which can be deployed in various types of network edges. It brings cloud computing capabilities and IT service environment at the edge of mobile network. This infrastructure can be implemented in several virtualization servers on different locations at the networks edge, such as Wireless Access Points (APs), LTE macro base stations (eNodeB), the Radio Network Controller (RNC) or the Radio Access Technology (3G/LTE/WLAN).

Deploying cloud services at the Edge of mobile networks will bring many advantages such as ultra-low latency and high bandwidth as well as real-time access to radio network information and location awareness. This will benefit the actual mobile infrastructure through optimization or new implementation preparing 5G. Also, new services and application deployment are emerging and it bring new horizon for 3rd party services providers through IoT, augmented reality, connected cars or intelligent video acceleration [29].



Figure 11 - Mobile Edge Computing architecture

The MEC architecture is based on a virtualized platform that enables application running on top of MEC servers, which can be deployed in various types of network edges. It brings cloud computing capabilities and IT service environment at the edge of mobile network. This infrastructure can be implemented in several virtualization servers on different locations at the networks edge, such as Wireless Access Points (APs), LTE macro base stations (eNodeB), the Radio Network Controller (RNC) or the Radio Access Technology (3G/LTE/WLAN).

Deploying cloud services at the Edge of mobile networks bring many advantages such as ultra-low latency and high bandwidth, as well as, real-time access to radio network information and location awareness. This will benefit the actual mobile infrastructure through optimization or new implementation preparing 5G. Also, new services and application deployment are emerging and it brings a new horizon for 3rd party services providers through IoT, augmented reality, connected cars or intelligent video acceleration [29].

For virtualized services, MEC deployment provides some key characteristic advantages like reducing costs of implementation, a standardize management and orchestration. Besides, MEC aims to reduce network stress by moving resources from cloud to mobile edge [40], with a fully virtualized system infrastructure in [56].

2.4. FRAMEWORKS DEPLOYMENTS

When this thesis started, ETSI was making great efforts to standardize the Mobile Edge technology. The first frameworks and applications were developed. Researchers from Industry and Academy tested some use cases through frameworks and applications.

In [57], the author presents Cloudlet Aided Cooperative Terminals Service Environment (CACTSE), a mobile content delivery service where mobile terminals are connected with each other via Service Manager (SM), which acts like a cloudlet module to improve the user experience.

The content is available through online or offline access, but it lacks of cache service.

In [58], Soyata presents an architecture based on mobile-cloudlet-cloud topology. The author proposes a Mobile Cloud Hybrid Architecture (MOCHA) as a framework for real time face recognition that gives the minimum response time. The author presents also CloudVision using that framework in order to decrease response time of a face detection and recognition task. In this framework, the Cloudlet can act as a buffer preventing heavy images to be transferred to the Cloud. It brings little benefits but high speed connection to the cloud is required and there is a space problem regarding number of faces to be used. In [59], the Author presents a Cloudlet based dictionary for mobile devices with support for translation of 6 languages, which is easily configurable and extensible. However, in order to present fast computation requires high processing power.

Koukoudidis proposes Pocket Cloudlet [60], a cloudlet framework that analyses and constructs a user and community behaviour model and tries to predict which data will be download in near future. The main goal is to identify the most popular contents and then download it to a cache storage. Data caching presents many challenges in determining exactly the balance between the data is required, update frequency and the storage available.

In the last years, some cloudlets architectures based on Virtual Machines were deployed in elastic cloud computing platforms like OpenStack. There exist also some differences regarding centralized or decentralized cloudlet management, and elastic or ad-hoc resources.

Carnegie Mellon University develops the cloudlet pioneering project, named Elijah Project, which is the initial extension to OpenStack++. This extension provide a cloudlet library based on a modified QEMU with integration into the open source OpenStack platform. A mesh cloud architecture is proposed in [55], which is composed of cloudlet, Internet cloud and wireless mesh networks. An experimental framework is designed in [45], in which private cloudlet and wireless mesh network is implemented. It is capable of establishing and maintaining mesh connectivity among multiple nodes automatically and is featured with adaptively and self-recovery in case of network failures. Instead of managing VMs for the deployment of a cloudlet system, Verbelen [61] propose a finer-grained cloudlet concept that offloads applications on the component level, without the need of sending a VM overlay. It also suggest that Cloudlets can be formed dynamically with any device in the LAN network that has available computing resources.

Abolfazli [62] proposes a dynamic cloudlet architecture consisting only of ad hoc cloudlet nodes, all of which are administered by a central service governor, a replicated supervisory entity that monitors and supervises computing augmentation entities.

2.5. SUMMARY

The last decade has seen a wide-range of new applications and services that require unprecedented high access speed and low latency experience, such as real-time online gaming, augmented reality and other cases presented in section 2.1. It drives the paradigm shift from the centralized Mobile Cloud Computing toward to the Edge.

The section 2.2. analyzes the requirements and key challenges for materializing 5G vision. Mobile Edge Networks are recognized as one of the key technologies necessary to reach next generation 5G and the natural development in the evolution of mobile BSs and the convergence of IT and telecommunication networking.

In Section 2.3., a background explains the convergence from Mobile Edge Computing to Mobile Edge Networks principal architectures proposed. Fog computing is initiated to address some challenges in meeting new requirements of IoTs, it provides high-performance, interoperability, and security in a multi-vendor fog computing-based ecosystem. MEC is recognized as one of the key technologies to meet 5G requirements, it enables an open RAN which can host third party innovative applications and content at the edge of the network. Cloudlets propose to address some challenges in mobile computing. Cloudlet provides new classes of mobile applications that are both compute-intensive and latency-sensitive in an open ecosystem based on cloudlets. In terms of comparison, the similarity between the three technologies is openness. An analysis is made to some frameworks and architectures that surged in the beginning of this study in order to choose the best-case scenarios under hardware and software disponibility.

3. THE MEC AND CLOUDLETS

The increasing improvements made in the mobile device area of sensing, connectivity, display and sound quality or computational capacity will lead to the development of new mobile applications. It allows a new perception of interactivity through image, voice motion or location. However, these new applications shall extinguish rapidly the limits of the mobiles devices. On the same way, these applications are pushing well beyond the cloud resources regarding the user interaction, since end-to-end latencies would be almost tens of milliseconds. This situation is not affordable and it results in the distraction or even worst the frustration of the users.

More than ever before, users wants to use applications in real-time with high definition characteristics everywhere and at any time. The human perception and cognition augments through the emerging of new smart mobile devices and applications.

At the same time, efforts made by the major contributors in order to standardize the concept of this new emerging technology, since an historical view to the framework and architecture view.

3.1. STANDARDIZATION

Major actors of the Telecommunication's area identify MEC as a key enabler for IoT and mission-critical, vertical solutions, and recognize as one of the key architectural concepts and technologies. The concept of MEC was defined by ETSI as a new technology that *“provides an IT service environment and cloud-computing capabilities at the edge of the mobile network, within the Radio Access Network (RAN) and in close proximity to mobile subscribers”* [63].

ETSI published a white paper on MEC, where it considered MEC as key emerging technology to be an important component of future generation networks [5].

In this section, an introduction is made to the recent standardization intentions from the industry. It analyzed the referenced MEC server framework as well as the technical challenges and requirements of MEC systems. Typical use scenarios of MEC were already discussed in Section 2.1.

3.1.1. NORMALIZATION EVOLUTION

ETSI has established an Industry Specification Group (ISG) on MEC to develop a standardized, open environment that shall allow efficient and seamless integration of third-party applications across multi-vendor platforms in December 2014.

Until January 2017, MEC ISG has released six specifications, one of which provides a glossary of terms related to the conceptual, architectural and functional elements of MEC [63]. Another specification specifies the technical requirements enabling interoperability and deployment and describes example use cases and their technical benefits [64]. A framework and a reference architecture was presented to enable mobile edge applications to run efficiently and seamlessly in a mobile network [65]. Moreover, the forth specification in MEC ISG introduces a number of service scenarios that would benefit from the MEC technology [66]. The specification of the Proof of Concept (PoC) framework defines a framework to coordinate and promote multi-vendor PoC projects illustrating key aspects of MEC technology [67].

ETSI has announced six different Mobile Edge Computing Proofs of Concept (MEC PoCs) in Sep. 2016, which were accepted in MEC World Congress in Munich and contribute to strengthen the strategic planning and decision-making of organizations, and help to identify which MEC solutions may be viable in the network.

The last specification describes various metrics which can be improved through deploying a service on a MEC platform, such as latency, energy efficiency, network throughput, system resource footprint and quality [68].

MEC ISG started 9 new studies related to MEC APIs, management interfaces and essential platform functionality. In addition, the MEC in an NFV environment is emerging on an end-to-end mobility. The NFV platform may be dedicated to MEC or shared with other network functions or applications. MEC exploit the NFV management and orchestration entities and interfaces as much as possible.

This confidence on MEC Technology stimulates all community and provide an acceleration on the standardization pace. By defining and standardizing key edge computing interfaces, ETSI ISG MEC eases the path to interoperability and removes this key obstacle towards a broad industry adoption of edge computing. It should be noted here that ETSI ISG MEC remains the only standardization group in this space.

Early in 2017, ETSI MEC ISG has decided that Mobile Edge Computing had to be renamed as Multi-access Edge Computing in order to reflect the growing interest in MEC from non-cellular operators [69]. This phase, know as MEC Phase 2, leverages on the industry acceptance of the first phase of specifications and is aimed at strengthening the engagement with developers and service providers, which are ultimately the stakeholders that exploit MEC for their value added product propositions.

The 3rd Generation Partnership Project (3GPP) shows a growing interest in including MEC into its 5G standard, and functionality supports for edge computing identified and reported in a recent technical specification document [70].

On July 2017, EST ISG have published 5 API specifications identified in scope for Phase 1 of work. These include specifications relating to the essential functionality of the application enablement platform (API framework), specific service-related APIs (Radio Network Information and Location Information) and management and orchestration-related [71 – 75].

In September 2017, ETSI released standard API for User Equipment (UE) application interface; it contains the specification for the lifecycle management of the user applications over the UE application interface [76].

One Month later, 3 more API were published. The first one is the specification for the user equipment-initiated operations platform management [77]. The second one specifies the necessary API with the data model and data format for Bandwidth Management services

[78]. The third specification released is for End to End Mobility Aspects [79]; it focuses on mobility support provided by MEC and presents use cases and end to end information flows to support UE and Application mobility.

More important, Phase 2 shall expand the applicability of standards from mobile to all types of access. Phase 2 defines also how MEC integrates with NFV and address significant new use cases, such as connected cars.

On February 2018, ETSI published 2 new standards: UE Identity API and Deployment of Mobile Edge Computing in an NFV environment [80, 81].

Phase 2 should also see an increased emphasis on the industry outreach with growing action to move towards adoption of that API by the key industry groups, certification and application developer outreach.

3.1.2. MEC FRAMEWORK

ETSI's MEC framework and reference architecture is defined in the Group Specifications (GS) MEC 003 [64], these group of specifications are known to be widely used as a reference architecture for many early MEC implementations.

The MEC framework proposed in Figure 12 identifies and groups the high-level functional entities in the system: network level, the MEC host level and the MEC system management.

The Network level entities comprising connectivity to local area networks, cellular networks and external networks such as Internet. A major objective is to extend this capabilities to non-cellular.

In the MEC host level, the MEC host sits along with its associated management subsystem. The MEC host is constituted by the platform and the virtualization infrastructure where the applications run.

In the MEC system level management retains the global view of the whole MEC system, i.e., the collection of MEC hosts and the associated management subsystem.

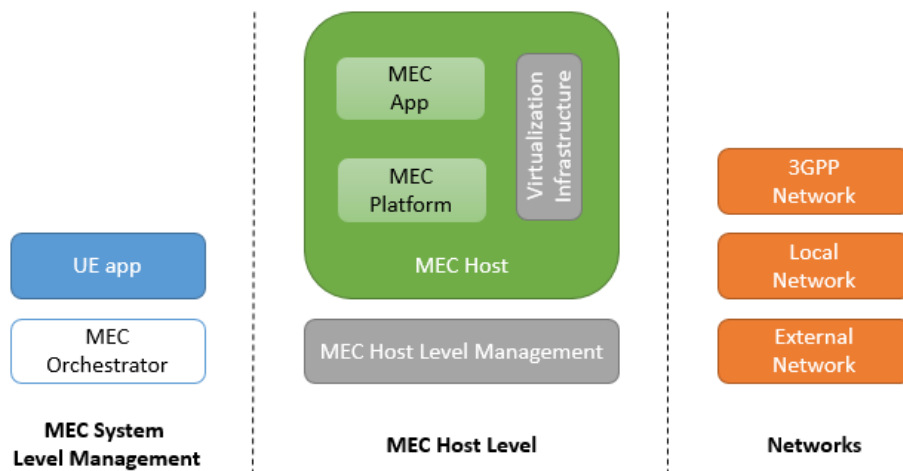


Figure 12 - MEC framework

3.1.3. MEC REFERENCE ARCHITECTURE

The MEC reference architecture (Figure 13) highlights the system level and host level components. Reference points in scope of MEC are represented by solid lines, while the reference points in scope of proprietary implementation or other Standards Developing Organizations (SDOs) are represented by dotted lines.

The **MEC host** is a logical construct that provides computing, storage and networking resources to the MEC applications enhancing the MEC platform and the virtualization infrastructure. The MEC platform send rules that are forwarded by an element inside the virtualization infrastructure, the data plane, which is also responsible for routing the traffic between the applications, services and the networks.

MEC host provides a virtualization infrastructure where **MEC applications** run as virtual machines. The applications may use MEC services present in the MEC platform or even provide them to the MEC platform and other applications.

The **MEC platform** holds essential functionalities that are required to run applications on the MEC host, which provides necessary features to discover, communicate, add and use MEC services.

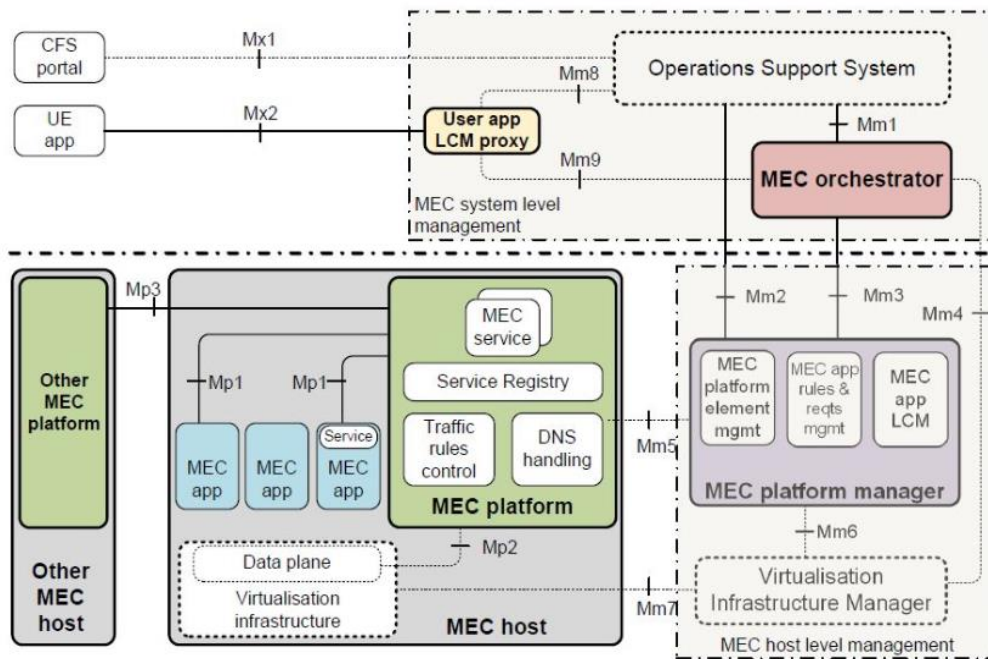


Figure 13 - MEC reference architecture [65]

In the Host level, the **MEC platform manager** consists of the MEC platform element management, the MEC application lifecycle management (LCM) and MEC application policy management functions. The LCM application is responsible for starting, finishing and relocating a MEC application instance. It provides some indications regarding events that occur in applications to the MEC orchestrator. The LCM application encompasses authorizations, traffic rules, DNS configurations and resolves issues when policies are in conflict.

The **Virtualization Infrastructure Manager (VIM)** is responsible of managing the virtualized resources for the MEC applications, like allocating and releasing virtualized computing, storage and network resources, therefore it has the Mm7 reference point towards the Virtualization Infrastructure for this purpose - OpenStack is a widely known example of a VIM [85].

The Management level is composed by the MEC Orchestrator, the Operations Support System and the User Application Lifecycle Management Proxy.

The **MEC Orchestrator** plays a central role as it has the visibility over the resources and capabilities of the entire MEC system. It is responsible to coordinate and control the

instantiation or resolve resource conflicts. The MEC Orchestrator manages the MEC applications and the associated procedures, such as integration, authentication, and validation of the policies related to them. It also checks if the proper requirements are set to the respective application.

The **Operations Support System (OSS)** is responsible for running the MEC applications in the proper location of the network. The Customer Facing Service portal (CFS) and the user equipment send requests to the OSS and to the orchestrator in order to instantiate and terminate applications. CFS provides an entry point for 3rd party services.

The **User Application Lifecycle Management Proxy** encompasses functions that allow the application clients to request services related to on-boarding, instantiation and termination of the applications.

3.2. CLOUDLET

A Cloudlet can be defined as “cloud on the box” with computing resources available for use by mobile users. During the execution of an application, the mobile device act as a client that offloads computation and data on the nearest cloudlet.

The specific problems that we are addressing are real time response, low latency, data management, scalability and resiliency. One of the existing problems, concerns with the cloud connection to remote server. To deal with that problem and achieve real-time response and low latency, the user applications can interoperate with the nearest cloudlet.

Another problem refers to the exponential growth of IoT smart devices and systems. It may result in the congestion of the backhaul network and scalability issues. The clouds need to forward, process, and store massive amounts of data generated by IoT devices.

Cloudlet architectures include caching mechanisms to filter and locally store essential information, avoiding unnecessary bandwidth consumption due to massive data transfers between IoT devices and the cloud servers.

When applications operate in real-time, the response time of the remote cloud server can result in application failures. The service becomes unaffordable due to the poor user

experience. Cloudlet architecture can be a resilient solution and offer on-premise and user control.

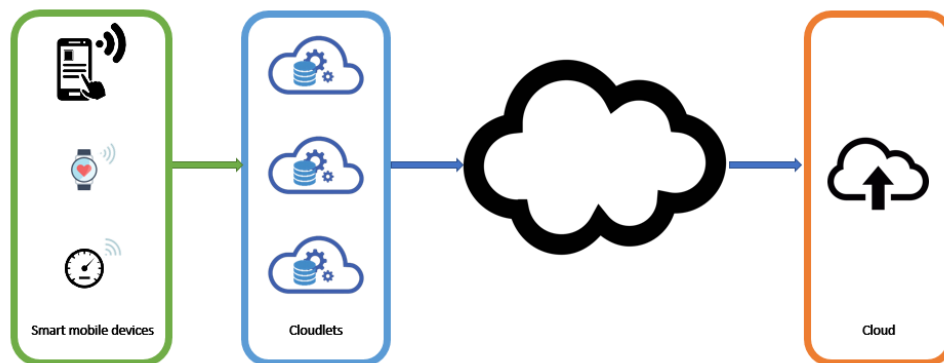


Figure 14 - Cloudlet architecture

Cloudlet architectures address the previously described challenge providing scalable deployment, management support and improved communications performance. Two types of elements define the cloudlet architecture: cloudlet host and mobile clients.

The philosophy of Cloudlet has been followed by the ETSI, leading to the standardization of MEC architecture [101][102][104].

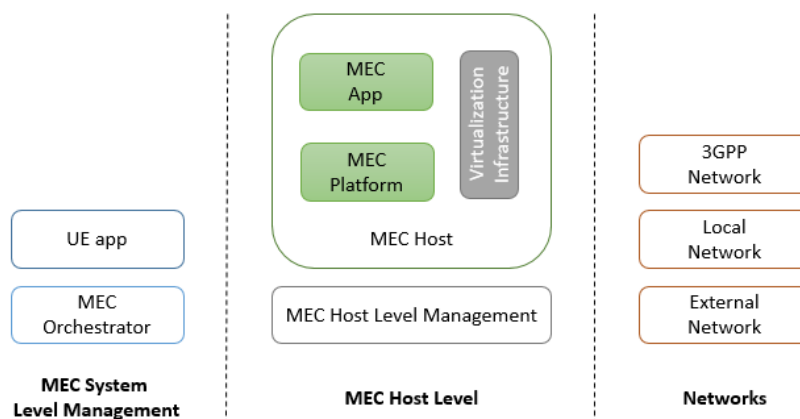


Figure 15 - Cloudlet framework proposed

The Cloudlet Framework used in this project is one of the base components of the ETSI framework, as show on Figures 15 and 16. The framework turns any network edge device in a MEC Host, containing a virtualization infrastructure and providing a platform for computing, storage and with network resources for MEC applications. The MEC platform has several functionalities that enable to run services and MEC Applications in a MEC Host,

with specific infrastructure virtualization conditions. The management of the MEC Host is performed through components that enable the configuration of each MEC host, platform or applications.

In terms of comparison with the ETSI model, cloudlets differs in some aspects regarding the virtualization infrastructure, the MEC Host level management and MEC Orchestrator, as can be seen in Figure 15 compared to Figure 16. Regarding the virtualization infrastructure and MEC Host level management, cloudlets only provide some basic functionalities in the MEC Host, while the ETSI model already defines a global Virtualization Infrastructure Manager (Open Platform for NFV - OPNFV) [97]. At last, the MEC orchestrator is not used in our architecture because this part is dependent on the ETSI MEC standardization that was not available at the moment of the setup of our demo implementation.

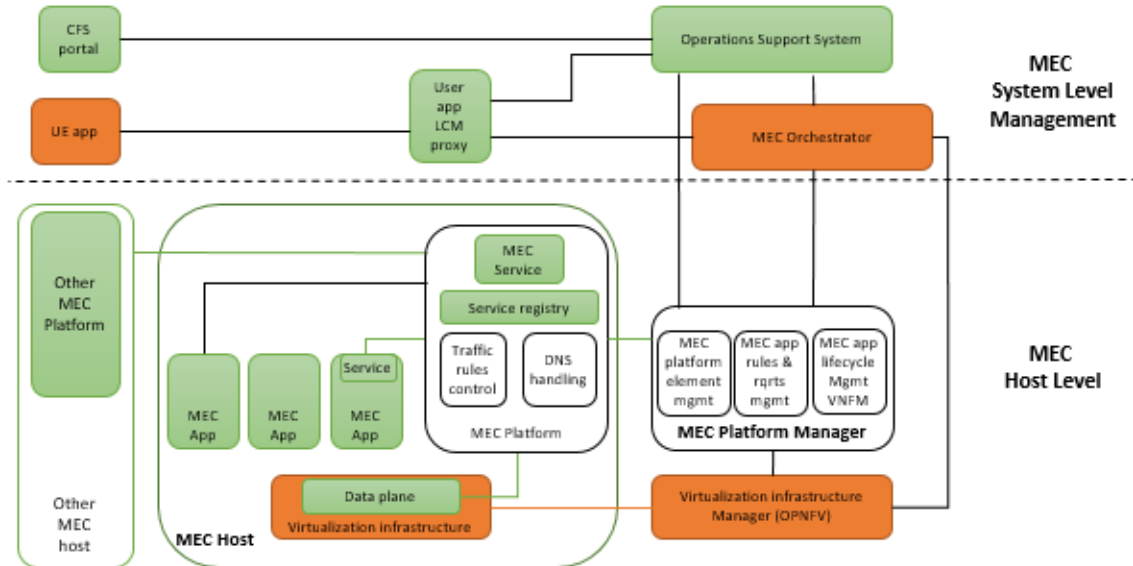


Figure 16 - Cloudlet vs ETSI MEC reference architecture

3.3. PROBLEM STATEMENT

In MEC architecture, the data management, processing and caching are performed directly at the network edge. In that manner, the throughput is reduced to avoid traffic in the backhaul network. Since this architecture is decentralized, the end-to-end latency is reduced and this increases the system resiliency by providing data redundancy, high availability, better quality of experience for the users.

The purpose of this work is to simulate MEC and cloud based solutions, that enable us to evaluate and quantify the real benefits of each architectural approach. As already discussed in the previous sections, it is claimed by several authors, that MEC leads to improvements in communications performance, such as ultra-low latency, high bandwidth and real-time access to radio network information and location awareness.

Furthermore, it is the main goal of ETSI [105]-[108], that MEC systems shall represent a solution regarding the 5G vision and new emerging applications with prerequisites latency of 1 millisecond.

This work aims to evaluate the key communication performance indicators of MEC and Cloud architectures in different applications scenarios. For this analysis, scenarios were considered that use real time and computing intensive task applications.

3.4. SUMMARY

Section 1 focuses on standardization efforts to develop new telecommunications frameworks and architectures. While ETSI is under initial challenge and gets the effort to regroup all community and purpose the first standards, this thesis aims to compare Mobile Edge Technology and Cloud-based solution and provide experimental evidence that recent emerging demands force a change in cloud computing architecture.

In section 2, this thesis analyzes the Cloudlet general architecture and summarizes each component and their use. After that, a synthesis introduces the problem statement about the use of the remote cloud servers. This project implements a solution to reduce excessive latency and network bandwidth at the edge of the network. Finally, we compared our proposed architecture with the model that ETSI seeks to standardize, and report the ongoing coordination efforts among the participants in standards development.

In the next chapter, this work presents the implementation of this architecture and its components. The purpose is to analyze the benefits of MEC through Cloudlet scenarios using new emerging applications.

4. CLOUDLET IMPLEMENTATION

The number of applications and mobile devices is increasing, and the prevision is to continue to grow. These new emerging applications required even more computation-intensive tasks and battery power. This obstacle prevents the achievement of the needed capabilities. Even if mobile devices have better capabilities, they still do not process the task demand. Some networks are often unreliable, thus the limited bandwidth can also be inconsistent. Resuming, the time to access remote cloud servers is unaffordable.

This project presents a anlyses an architecture that challenges all these obstacles. At the Edge, the MEC architecture proposed aims to be discoverable. Stateless servers can run one or more Virtual Machines (VMs) on which mobile devices can offload extensive computation, as presented in Figure 17. This architecture enhances processing capacity, conserving battery and proven to solve the characteristic bottleneck problems related to cloud technologies.

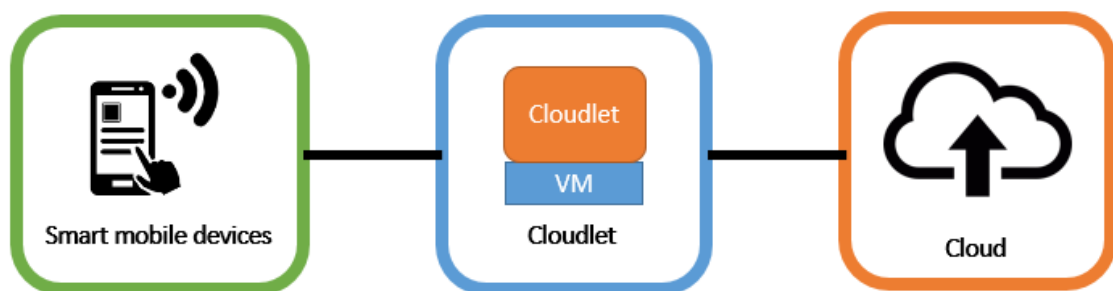


Figure 17 - Three-tier architecture for code offload

4.1. CLOUDLET ARCHITECTURE

This project offers a code offload solution at the network edge for smart mobile devices that exploit cloudlets. Figure 18 presents the major components of the Cloudlet architecture, which are the mobile client and the Cloudlet Host.

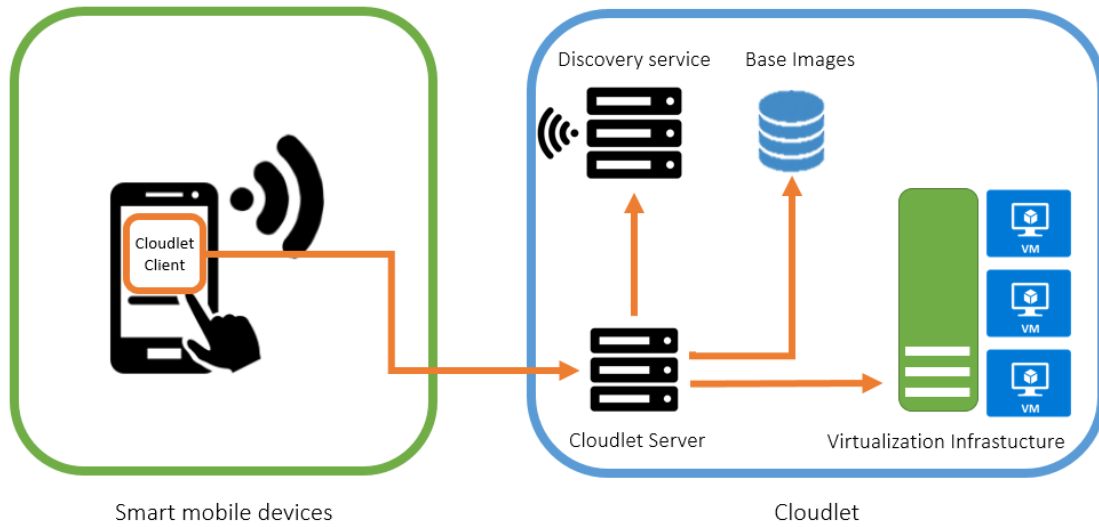


Figure 18 - VM-based cloudlet architecture

The Cloudlet Host used is a VM-based cloudlet architecture [12] [82] [83] [84]. There are similarities with Cloud data centers. It's Virtual Machines have similar requirements, such as a wide range of computations, programming language, operative system, and dynamic resources allocation.

The Cloudlet Host is a physical server hosting a discovery service that broadcasts the Cloudlet IP address and port to allow mobile devices to find it. It contains the Base VM images used to synthesize the VMs. It hosts a Cloudlet server that performs the synthesis of the VM, handles the code offload through application overlays or starts guest VM instances. Finally, the Cloudlet Host contains a VM manager that acts as a host for guest VM instances and stores the computation components of mobile apps.

The Mobile client is a smart mobile device that hosts a Cloudlet client application responsible to discover cloudlets and uploads the application overlays to the Cloudlets. It

also contains mobile applications that operate as clients of the Cloudlets servers. The mobile client stores an application overlay of each Cloudlet-ready application that a user wants to execute and for which computation offloading is appropriate. In the Cloudlet, the same Base VM image generates each application Base VM image.

This VM-based Cloudlet architecture presents some important differences regarding the Cloud, such as the rapid provisioning, the VM handoff, and Cloudlet discovery.

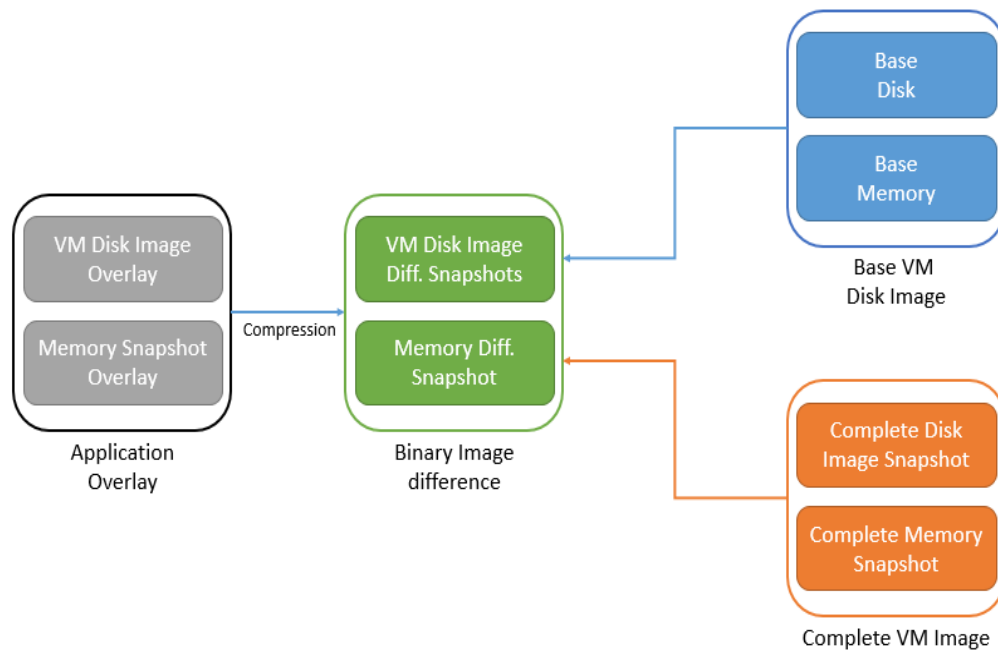


Figure 19 - VM overlay creation

First, cloud data centers have stored most of the VM images, presenting an optimization of the provision of VMs. So, if a user launches a new image, the cloud does not meet fast launch requirements. Cloudlet needs to be agile with VM provisioning since a mobile user needs dynamic association and response time.

The second matter regards live VM migration across cloudlets. The user mobility across the localization occurs if there is a transfer of offloaded services from a source cloudlet to a destination cloudlet.

Finally, a dynamic Cloudlet discovery is essential when a mobile client needs to discover and associate a particular cloudlet among many candidates before the provisioning.

4.1.1. VM SYNTHESIS

In a VM-based Cloudlet architecture, VM synthesis is crucial to provide features like rapid provisioning and VM handover. Operating System (OS), libraries, and packages are the principal components of a VM image. In comparison, the needed user application part is tiny. If a Base VM already exists in the cloudlet, it is only necessary to transfer the difference part, which is the VM overlay. The VM synthesis is the method used to provision the cloudlets using VM overlays.

Figure 19 presents a VM overlay creation from a Base VM image. Users can generate Base VM images from popular OS builds like Linux or Windows. When a pause occurs of the booted image, the snapshot of the VM disk image and the memory snapshot creates the base disk and base memory. The user needs to resume the instance, install and configure all necessary components of the back-end server-side application. Finally, the user has to launch the back-end server again and then pause it to perform the snapshot of the resulting disk image and memory of the final VM image with the back-end server. The resulting application overlay generates the final VM image and the base VM image using xdelta3 and LZMA the compression.

The direct provision of the mobile back-end application in the Cloudlet platform performs the VM synthesis. Also, an overlay delivery from the cloud or even from the storage on the mobile devices can perform a VM Synthesis. In the delivery's case of the overlay, it will decompress in a base image to generate a launch VM that will create a VM instance. After that, the mobile device can perform many actions with the instance.

4.1.2. LIMITATIONS

The architecture focuses on a cloudlet platform that provides a data processing system where the application's data can be cached, processed and aggregated. Given the frame time of this thesis project, the standardization of mobile edge computing was still a mirage and it was still trying to regroup all the market players. The first platforms and applications developed were still under test and the changes were constant. This work does not address the inter-cloudlet communication and multiple cloudlet integrations with a cloud discovery service.

4.2. OPENSTACK

Cloudlet architecture presents different technical challenges. One of them is the cloudlet deployment. In the previous subsections, a summary presents the composition of the Cloudlet architecture and framework. In this subsection, we will introduce the proposed Cloudlet platform to perform our practical analysis through the offloading of mobile applications. OpenStack is a free and open-source cloud-computing platform that offers the possibility to establish and test new emerging architecture or ecosystem.

OpenStack++ is a particular API extension on the OpenStack platform that implements the deployment of the Cloudlet platform.

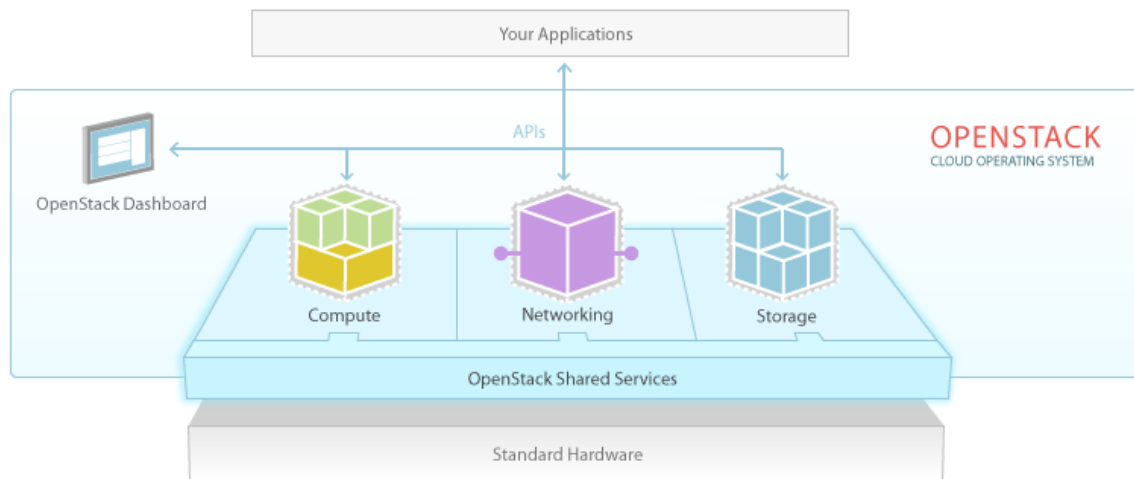


Figure 20 - OpenStack software diagram [85]

OpenStack is a well-known free and widely diffused open-source Infrastructure-as-a-Service (IaaS) software platform for private or public cloud computing. As presented in Figure 20, it provides many services that interact with each other to deliver the full feature set and to manage computation, storage, and networking resources to supply dynamic allocation of VMs. Users can manage this platform through a web-based dashboard, a command-line tool, or a RESTful API.

OpenStack is a project started in 2010 as a joint project of Rackspace Hosting and NASA and managed by OpenStack Foundation. Since its founding, it has seen wide industry endorsement and now numbers over one hundred supporters, including many of the

industry's largest organizations, such as AT&T, Rackspace, Cisco, SUSE, IBM, Juniper, Yahoo, HP, Intel, Red Hat, Canonical, Yahoo, Dell, Vmware.

It fulfills the cloud: massive scalability and simplicity of implementation. OpenStack is highly configurable, i.e. the user can choose whether to implement each one of the several services offered by the software. The application programming interface tool (API) allows the user to configure each component easily. Therefore, OpenStack is a flexible tool able to work along with other software.

Another reason to adopt OpenStack is that it supports different hypervisors (Xen, VMware or kernel-based virtual machine (KVM) for instance) and several virtualization technologies (such as bare metal or high-performance computing).

4.2.1. SERVICES

The OpenStack community has collaboratively defined the key components of the “core” of OpenStack, which are distributed as a part of the system and officially maintained by the OpenStack community. The conceptual OpenStack architecture is illustrated in Figure 21.

Nova is the service responsible for computing behind OpenStack. It deploys and manages virtual machines and other instances to handle computing tasks.

Swift is a storage system for objects and files, based on a unique identifier to refer to a file or piece of information. OpenStack provides an easy scaling function as it decides where to store the information and backups in case of machine or network connection failure.

Cinder is the block storage component that controls the method to access specific locations on a disk drive. This file access method might be important in scenarios in which data access speed is of most importance.

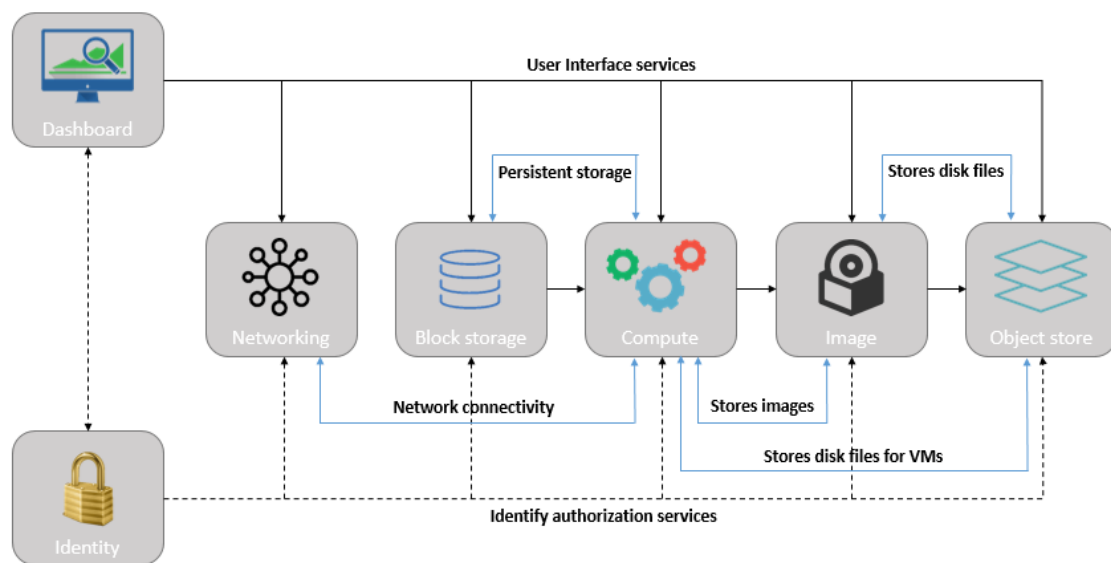


Figure 21 - Conceptual OpenStack architecture

Neutron provides the networking capability for OpenStack and ensures that the components can communicate with each other, quickly and efficiently.

Horizon is the dashboard behind OpenStack. This graphical interface allows developers to access all the components of OpenStack individually through an API. The dashboard also provides a system administrator access to monitor and manage the cloud.

Keystone provides identity services for OpenStack. It is essentially a central list of all the users of the OpenStack cloud, mapped against all the services provided by the cloud, which they have permission to use.

Glance is the service responsible for providing images to OpenStack.

Ceilometer provides telemetry services, which allow the cloud to provide billing services to individual users of the cloud. It also keeps a verifiable count of each user's system usage of each of the various components of an OpenStack cloud. Think metering and usage reporting.

Heat is the orchestration component of OpenStack that manages the infrastructure needed to run a cloud service. It allows developers to store the requirements of a cloud application in a file that defines what resources are necessary for that application.

4.3. OPENSTACK EXTENSION - OPENSTACK ++

OpenStack++ is the open-source extension API that enables the Openstack infrastructure in Cloudlet. The Elijah project was born at Carnegie Mellon University [53] to provide a Cloudlet extension that specifies the MEC platform idealized by the ETSI model, as shown in Figure 22. Elijah is a MEC-oriented extension of OpenStack with a relevant and growing community of MEC developers working on top of it. Some of the major actors started the Open Edge Computing Initiative (OEC) driving the development of the ecosystem around Edge Computing. Some of such actors are: Carnegie Mellon University, Intel, Nokia, Crown Castle, Vodafone, T-Mobile, and NTT [56].

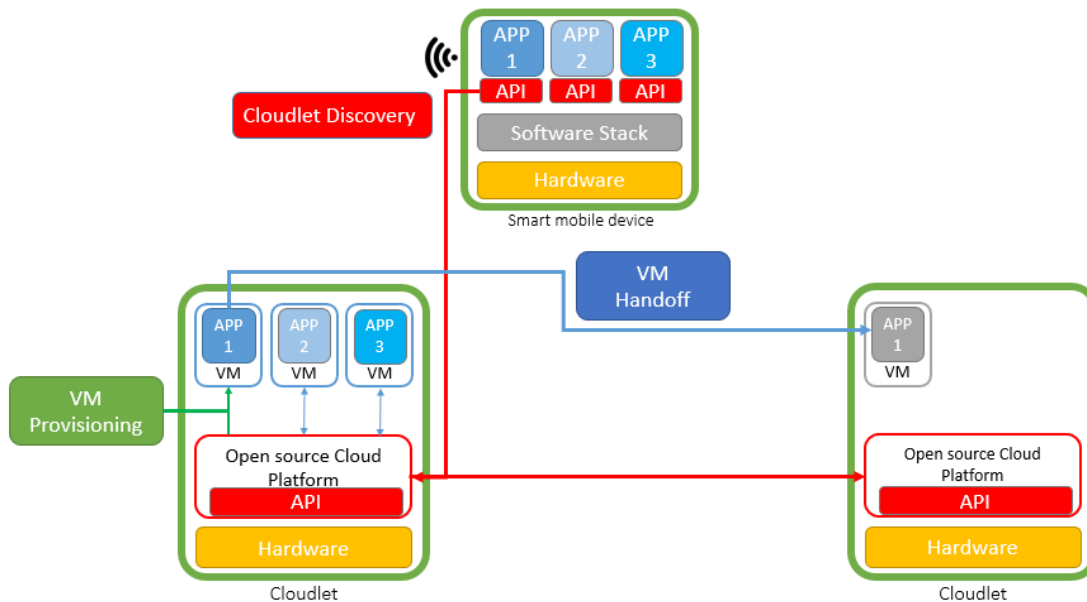


Figure 22 - Openstack++ cloudlet platform

OpenStack++ has some functionalities similar to OpenStack, and it also offers the possibility to add or remove certain features to the platform through the customization of the certain files in specific APIs. Figure 23 represents the files associated with the respective OpenStack API, which are *cloudlet.py*, *cloudlet_api.py*, *cloudlet_manager.py* and *cloudlet_driver.py*.

The Cloudlet configuration files cannot be directly changed into the respective folders, but it requires changing the original OpenStack classes and files. OpenStack is a complex open-source end-to-end cloud computing platform, that contains typical functionalities required to operate cloud computing. However, as the Openstack platform frequently makes updates, the implementation and the maintenance of this platform is not trivial.

4.3.1. PLATFORM SETUP

OpenStack++ implements the Cloudlet on the most OpenStack stable release version (Kilo). Implementing the platform OpenStack Kilo and the OpenStack++ extension was the least arduous, as much as this project was still in the beginning and suffered from constant changes from day to day.

We deploy the installation of the Elijah project using the DevStack method, which is a set of utilities scripts that aim to deploy quickly an OpenStack cloud from GitHub source trees in a clean Ubuntu or Fedora environment. Therefore, we used a workstation with Operative System Ubuntu 14.04 LTS 64 bits.

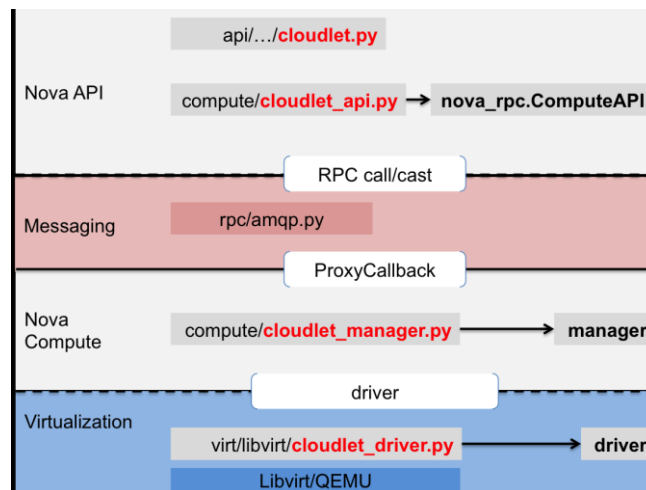


Figure 23 - Cloudlet API call hierarchy [85]

Three phases comprise the intended installation process of all platform in one single machine: the installation of the cloudlet library, the installation of the OpenStack platform through DevStack and at last the extension OpenStack++ [85]. The undergoing test of this project installation process shown several implementation errors, misconfigurations and download links failures, because the project was still in development and improvement phase. So, many bugs, problems with libraries and transfer links were detected and solved during the setup process. The Cloudlet installation was involved in many difficulties and complexity.

The fabric script for the Cloudlet installation had a quirk. Before the installation, it asked for the root password when it was designed to run as a local user.

Before starting the second process that installs the OpenStack cluster as a single cloudlet, it is necessary to configure the *local.conf* and *stacks* files. We configured DevStack through the *local.conf* file and changed all custom and local settings, such as services admin password, and the cluster controller settings.

It is possible to move the network ranges away from the local network or also set the host IP address if detection is unreliable. We configured the *stackrc* file with the version Kilo of OpenStack.

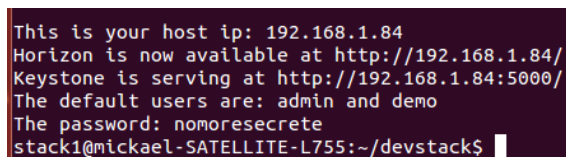
Then, we proceeded with the installation of OpenStack performed through DevStack and finally the setup of the extension part. Since many errors appeared due to:

- Different pip library versions used for each API, lead to bugs and the installation became hazardous with many installs and uninstall during the process;
- GitHub repositories had recurrent development changes, without provision of a stable version;
- Fatal errors during command line installation, because downloads of certain libraries from dead links, break down the installation process.

After some weeks of intense research and active participation with Open Edge Computing members, we finished the setup of this project platform. Figure 24 presents the last setup part. While starting the cloudlet platform properly without error, it is also necessary to instantiate every time:

- the authentication component Keystone;
- the Apache server to launch the dashboard;
- the volume manager Cinder that does not initialize the driver by itself;
- the shared service to manage token authentication, nova-consoleauth.

It was possible to reach the User Interface and access the Cloudlet Dashboard and all functionalities shown in Figure 25.



```
This is your host ip: 192.168.1.84
Horizon is now available at http://192.168.1.84/
Keystone is serving at http://192.168.1.84:5000/
The default users are: admin and demo
The password: nomoresecrete
stack1@mickael-SATELLITE-L755:~/devstack$
```

Figure 24 - OpenStack++ final configuration setup

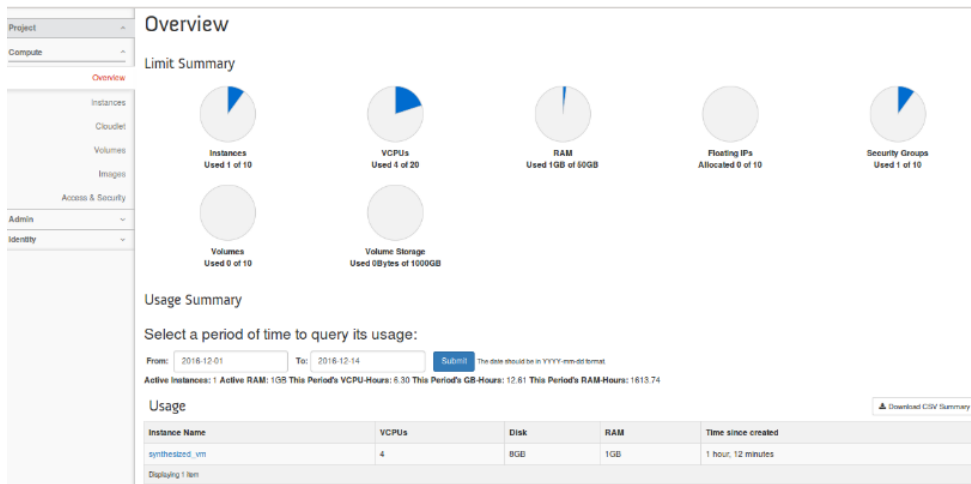
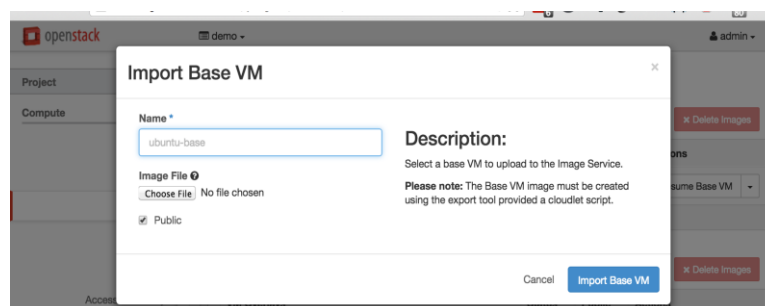


Figure 25 - OpenStack Dashboard

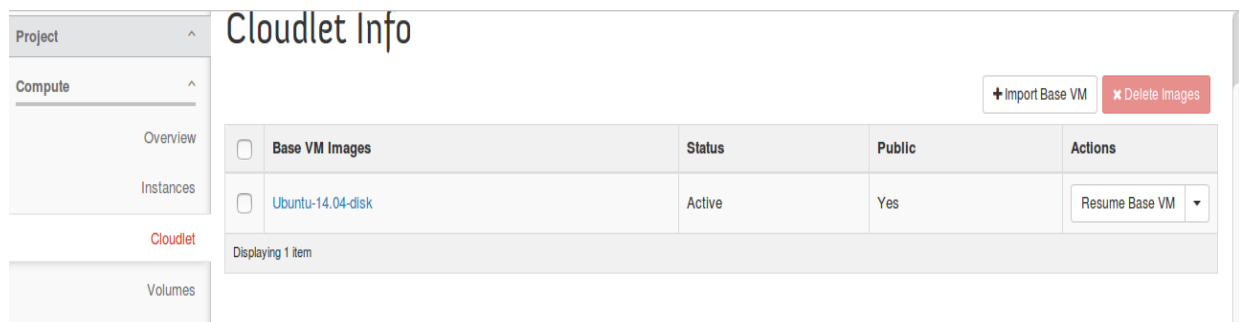
4.3.2. OPENSTACK++ FEATURES

The most relevant OpenStack++ features used in this project, are base image import, base image resume, overlay creation, and VM synthesis.

The function “**Import Base VM**” allows the users to load in advance the base image into Glance storage to build each VM. Figure 26 presents this online task. Figure 27 shows each base image is a compressed file that contains 4 files: a base disk image with the related hash value list and a memory snapshot with the related hash value list; is_cloudlet flag that shows that is not a standard cloud image; libvirt configuration with the metadata that shows the characteristics of the VM generated with the base image. The Elijah command to import a base image is cloudlet import-base that decompresses the base image and stores it into the Cloudlet database with the assignment of a unique identified hash.



(a) Import Base VM



(b) Base VM imported appearance in dashboard

Figure 26 - Import Base VM Image Process

Name	Ubuntu-14.04-disk
ID	acf2c2b6-8f5d-4fe0-bb4e-be894f535430
Owner	212445136b10495ea719d5846a29a653
Status	Active
Public	Yes
Protected	No
Checksum	0fa757b36ced50d5b9526827017781eb
Created	Dec. 9, 2016, midnight
Updated	Dec. 27, 2016, 3:16 p.m.

Specs	
Size	8.0 GB
Container Format	BARE
Disk Format	RAW
Min Disk	8GB
Min RAM	1GB

a) Base VM disk image file metadata

```

stack1@mickael-SATELLITE-L755:~$ glance --os-username=admin --os-password=nomoresecrete --os-tenant
00/v2.0 image-show Ubuntu-14.04-diskhash
+-----+
| Property | Value |
+-----+
| Property 'base_sha256_uuid' | abda52a61692094b3b7d45c9647d022f5e297d1b788679eb93735374007576b8 |
| Property 'cloudlet_type' | cloudlet_base_disk_hash |
| Property 'image_location' | snapshot |
| Property 'image_type' | snapshot |
| Property 'is_cloudlet' | True |
| checksum | dc38f9ba709ec10fa22f83da76bc9dac |
| container_format | bare |
| created_at | 2016-12-09T00:00:56.000000 |
| deleted | False |
| disk_format | raw |
| id | 3bc774e1-92d1-4564-8563-7dbf9030e8d4 |
| is_public | True |
| min_disk | 8 |
| min_ram | 1024 |
| name | Ubuntu-14.04-diskhash |
| owner | 212445136b10495ea719d5846a29a653 |
| protected | False |
| size | 118018164 |
| status | active |
| updated_at | 2016-12-09T00:01:00.000000 |
+-----+

```

b) Base VM memory snapshot metadata

Figure 27 - Base VM files metadata

The function **Resume Base VM** is still an offline operation and usually follows the import base image, as shown in Figure 28. To resume a base image, the Cloudlet platform uses a cloudlet hypervisor driver class, called *CloudletDriver*, that inherited the original LibvirtDriver and check if the metadata associated to the virtual disk image base has the *is_cloudlet* flag. Here, the driver resumes the base VM from the snapshot, rather than boots a new VM instance. Usually, the first time it takes a long time to resume a base image, in the order of a few minutes in relation to the hardware capability of the host, it verifies all permission, quota or other resource availability. The offline task made in advance prepares the MEC node before receiving the users' requests. In this way, OpenStack++ imports the base image into the cache of the compute node, thus, it does not slow down the system. Users are not significantly perceiving by the users for further base image resumes. At the end of this operation, there is a VM ready to execute the service.

Cloudlet Resume Base VM

Base VM Info

Image

ubuntu-12.04-disk

Instance Name

resumed_vm

Security Groups

default

Flavor

cloudlet-flavor-ubuntu-12.04

Description

We resume the instance from the glance image server.

If the Base VM has not cached, this process can take a few minutes.

Cancel

Launch

(a) Resume Base VM Setup

+

Start VM Synthesis

<input type="checkbox"/>	Instance Name	Type	IP Address	Size	Status	Task	Power State	Actions
<input type="checkbox"/>	resumed_vm	VM instance	10.11.12.3	cloudlet-flavor-Ubuntu-14.04 1GB RAM 4 VCPU 8GB Disk	Build	<div>Spawning</div>	No State	<div>Terminate Instance</div>

(b) Resume Base VM handling process

<input type="checkbox"/>	Instance Name	Type	IP Address	Size	Status	Task	Power State	Actions
<input type="checkbox"/>	resumed_vm	VM instance	10.11.12.3	cloudlet-flavor-Ubuntu-14.04 1GB RAM 4 VCPU 8GB Disk	Active	None	Running	<div>Create VM overlay</div>

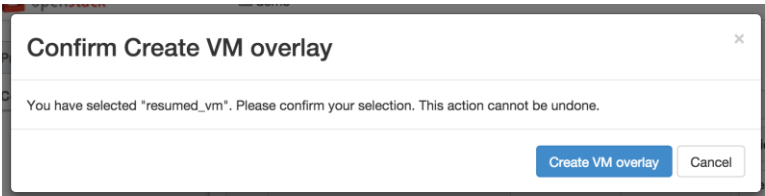
(c) Resume Base VM final process

Figure 28 - Resume Base VM creation process

At this operation, a developer prepares a back-end server at the middleware layer and typically this phase includes: preparing dependent libraries, downloading and setting executable binaries, and changing OS and system configurations.

Figure 29 presents the feature **Create VM Overlay** aims to create a minimal VM overlay starting from a resumed or running instance and then compress and save the VM overlay in Glance storage for later download. VM overlay is able to create snapshots used later to resume the VM from a specific moment, by containing the delta between the client VM and the base image VM. It contains all the changes we need to add on the base VM to reproduce the client VM environment at the moment of the migration. It adds this functionality with the extensions mechanism, defining a new virtualization driver `CloudletDriver` class that

inherits *nova_rpc.ComputeAPI*. The Elijah command to create a customized VM based on top of the base VM is a cloudlet overlay.



(a) Create VM overlay process – Instance Setup

✖ Delete Images

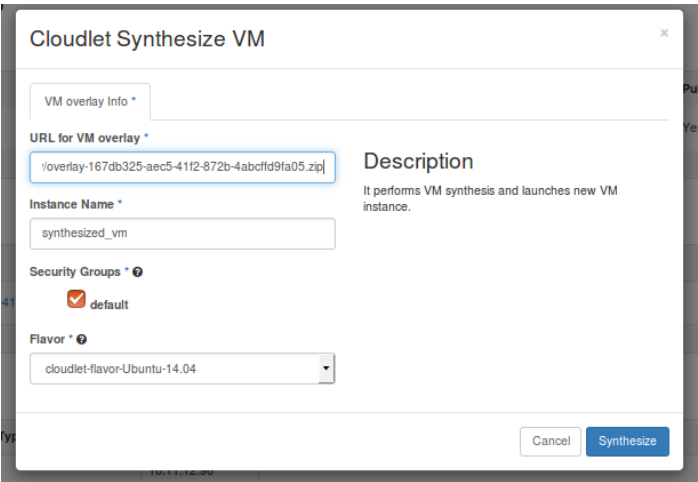
<input type="checkbox"/>	VM Overlays	Status	Public	Actions
overlay-167db325-aec5-41f2-872b-4abcf9fa05	Active	Yes	<div>Download VM overlay ▾</div>	

Displaying 1 item

(b) Create VM overlay process – Final state

Figure 29 - Overlay creation process

As previously explained, it was not possible to implement VM Handoff. The function **VM synthesis** launches a new VM instance to the OpenStack cluster, a process known as VM Provisioning. It uses an HTTP POST message with the `overlay_ulr` parameter and CloudletDriver hypervisor driver handles this message and manages the VM spawning methods to perform VM synthesis using the VM overlay and the VM base image. The commands `synthesis_server` for the server invokes the synthesis mechanism and listens locally and `synthesis_client` with the specification of the server IP and the overlay URL as presented in Figure 30.



(a) VM provisioning setup

<input type="checkbox"/>	Instance Name	Type	IP Address	Size	Status	Task	Power State	Actions
<input type="checkbox"/>	synthesized_vm5	VM instance	10.11.12.19	cloudlet-flavor-Ubuntu-14.04 1GB RAM 4 VCPU 8GB Disk	Active	None	Running	Terminate Instance
<input type="checkbox"/>	synthesized_vm4	Provisioned VM	10.11.12.10	cloudlet-flavor-Ubuntu-14.04 1GB RAM 4 VCPU 8GB Disk	Active	None	No State	Terminate Instance
<input type="checkbox"/>	synthesized_vm3	Provisioned VM	10.11.12.7	cloudlet-flavor-Ubuntu-14.04 1GB RAM 4 VCPU 8GB Disk	Active	None	No State	Terminate Instance

(b) VM provisioning final state

Figure 30 - Cloudlet VM Synthesis creation process

Figure 31 illustrates the **VM handoff** interface which enables to migrate VMs between different OpenStack nodes. Since it involves two independent nodes, it is necessary that the user has permissions to access them and call the APIs. The command to execute the handoff uses a Python file, called *cloudlet_client*, which requires the UUID of the VM to migrate and the credentials to access both OpenStack nodes. It is possible to perform VM handoff only if the VM is synthesized.

Handoff VM instance

Keystone endpoint for destination OpenStack *

mist.elijah.cs.cmu.edu:5000

Destination Account *

admin

Destination Password *

.....

Destination Tenant *

demo

Instance Name at the destination *

handoff-vm

Description:

VM handoff will migrate the VM instance to other OpenStack cluster. This process is optimized for reducing migration time by applying various techniques covered on <http://reports-archive.adm.cs.cmu.edu/anon/2015/CMU-CS-15-113.pdf>.

Although it **does not save any credential information**, please use command line client that accepts auth-token instead of account/password if you don't want to pass account information.

Cancel

Handoff VM instance

Figure 31 - VM Instance Handoff setup creation

5. PERFORMANCE TESTS SETUP

The performance evaluation measures the latency of the system, as the time interval between the user's request and the system response. This includes bottlenecks in the network, wireless access collisions, optical fiber delays, hardware, and operating system latency.

Figure 32 presents the architecture of our testing workbench. The environment defined enables the performance measurement of the cloudlet and cloud solutions. This work implements two use cases with several testing cycles. Finally, an analysis compares the data collected.

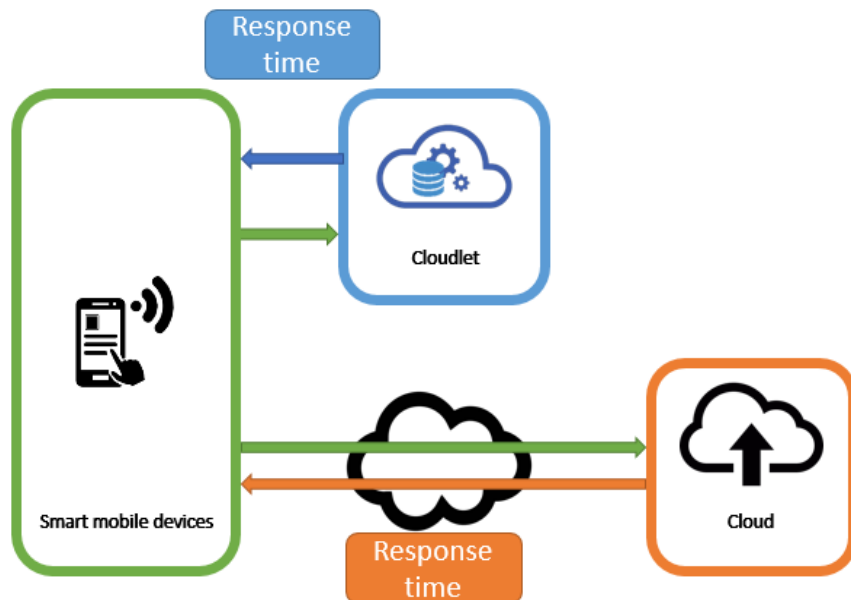


Figure 32 - Performance evaluation diagram

5.1. CLOUD PLATFORM

In order to evaluate the difference between cloud and cloudlet regarding key features, such as latency or user quality of experience, a cloud platform service was selected. The cloud platforms that offer better capabilities around flexible compute, storage and networking are Amazon Web Services (AWS), Microsoft Azure and Google Cloud Platform. They all share the common elements of a public cloud: self-service and instant provisioning, auto-scaling, plus security, compliance and identity management features. It is possible to launch Virtual Machines with many types of OS, such as Windows 7, Ubuntu 14.05.

Our test scenarios were implemented on the AWS platform, which provides a range of functionalities, a list of tools and services. Moreover, one of the our test scenarios is based on FaceSwap application, which only has an Android server disk image on Amazon EC2. This server application can launch instances with many configuration types, such as different number of cores and RAM. This feature was crucial to the choice of the cloud platform.

AWS Educate is Amazon's global initiative to provide students with the resources needed to hands-on access to AWS technology, training resources and to test free tier experiments. It is possible to launch AWS Virtual images freely with 1 CPU core for some considerable timeline and data transfer. The two applications used in the scenarios have a recommended processing of more than 1 CPU core, so several test setups with more CPU cores were also used. Unfortunately, this configuration is only part of the test to perform and, for other testing scenarios with extended virtual machine resources, we had to pay to perform tests that require those conditions.

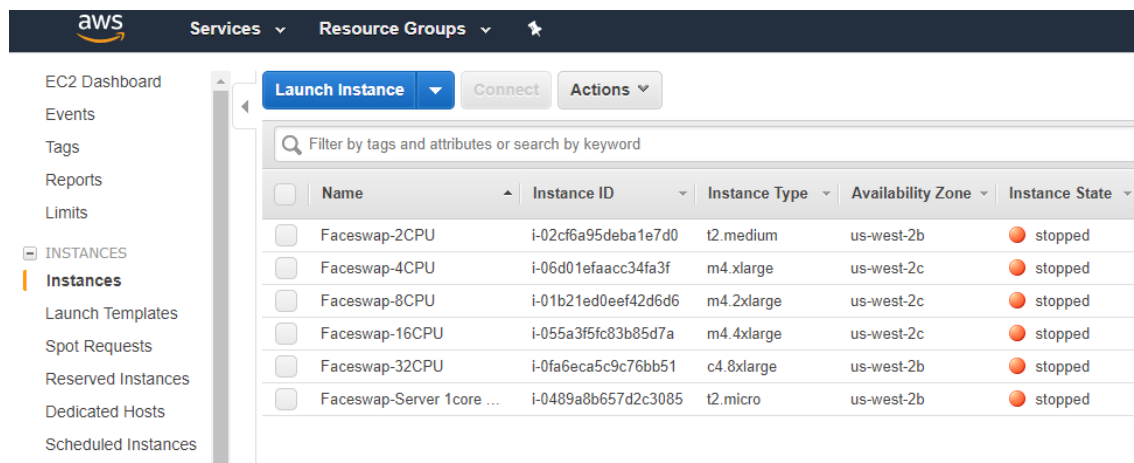
5.1.1. AWS FUNCTIONALITIES

AWS offers a Platform as a Service (PaaS), IaaS, serverless computing and much more, with over 70 different services.

Amazon VPC is the Virtual Private Cloud and allows building virtualized private networks inside of AWS, with subnets, NAT gateways, VPN connections, routing tables, security groups.

Amazon Elastic Compute Cloud (Amazon EC2) is a web-based service that allows businesses to run application programs in the Amazon Web Services (AWS) public cloud. Amazon EC2 allows a developer to spin up virtual machines (VM), which provide compute capacity for IT projects and cloud workloads that run with global AWS datacentres, as presented in Figure 33.

The Amazon EC2 web interface provides a scalable service as it allows the user to increase or decrease instance capacity within minutes. A developer can define auto-scaling police to scale instances automatically or manage multiple instances at once.



The screenshot shows the AWS Management Console for the EC2 service. The left sidebar contains navigation links: EC2 Dashboard, Events, Tags, Reports, Limits, INSTANCES (expanded), Instances (selected), Launch Templates, Spot Requests, Reserved Instances, Dedicated Hosts, and Scheduled Instances. The main content area has a 'Launch Instance' button, a 'Connect' button, and an 'Actions' dropdown. Below these is a search bar and a table of instances.

	Name	Instance ID	Instance Type	Availability Zone	Instance State
<input type="checkbox"/>	Faceswap-2CPU	i-02cf6a95deba1e7d0	t2.medium	us-west-2b	stopped
<input type="checkbox"/>	Faceswap-4CPU	i-06d01efaacc34fa3f	m4.xlarge	us-west-2c	stopped
<input type="checkbox"/>	Faceswap-8CPU	i-01b21ed0eef42d6d6	m4.2xlarge	us-west-2c	stopped
<input type="checkbox"/>	Faceswap-16CPU	i-055a3f5fc83b85d7a	m4.4xlarge	us-west-2c	stopped
<input type="checkbox"/>	Faceswap-32CPU	i-0fa6eca5c9c76bb51	c4.8xlarge	us-west-2b	stopped
<input type="checkbox"/>	Faceswap-Server 1core ...	i-0489a8b657d2c3085	t2.micro	us-west-2b	stopped

Figure 33 - AWS EC2 Dashboard

To use EC2, developers create an Amazon Machine Image (AMI) containing an operating system, application programs, and configuration settings. The Amazon Simple Storage Service (Amazon S3) uploads the AMI and registers it with Amazon EC2, creating an AMI identifier. Once done, the subscriber can restart virtual machines on an as-needed basis.

Data only remains on an EC2 instance while it is running, but developers can use an Amazon Elastic Block Store volume for an extra level of durability and Amazon S3 for EC2 data backup. VM Import/Export enables to import on-premises virtual machine images to Amazon EC2 for launching their instances.

5.2. TESTBENCH SCENARIO

This work conducted all experiments using the configuration shown in Figure 34. We create a testing workbench, which is a fixed development environment that is reproducible and portable. This environment allows us to measure the performance of the cloudlet and the cloud. In the two use cases, an Android client application runs on a smartphone, and servers are run on the cloudlet and in the cloud.

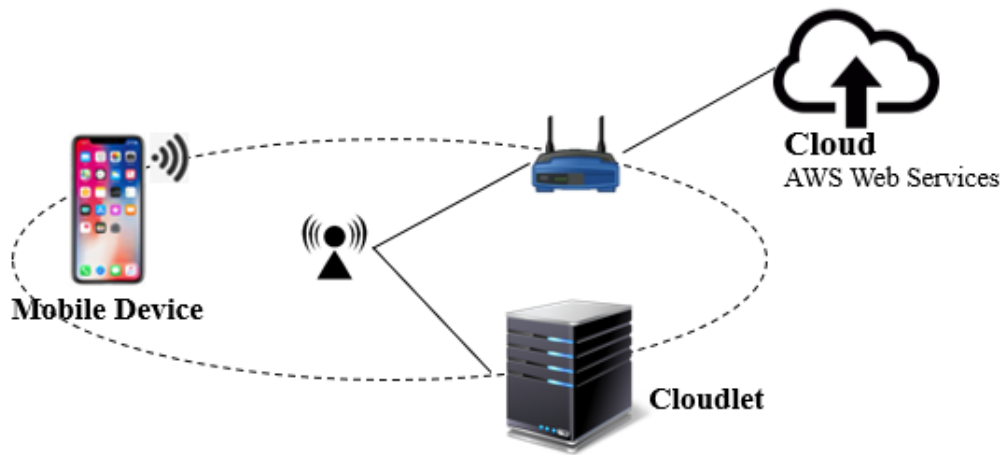


Figure 34 - Evaluation infrastructure setup

The smartphone used to test the client is a Samsung S8. Table 3 presents the main characteristics of this device [87].

Table 3 - Smartphone specification

Galaxy S8 specifications	
Processor	CPU: Octa-core (2.3GHz Quad + 1.7GHz Quad)
Display	Size: 5.8" Quad HD+ Super AMOLED (2960x1440)
OS	Android 7.0
Camera	Main Camera: 12.0 MP, Front Camera: 8.0 MP
Memory	RAM size: 4.0 GB, ROM size: 64.0 GB
Network/Bearer	3G, 4G
Connectivity	GPS, Glonass, Wi-Fi 802.11 a/b/g/n/ac 2.4+5GHz, Bluetooth 5.0

Sensors	Accelerometer, Fingerprint Sensor, Gyro Sensor, Geomagnetic Sensor, Light Sensor, Proximity Sensor
Physical specification	Dimension (mm) 148.9 x 68.1 x 8.0, Weight (g) 155

The smartphone uses an access point Wi-Fi connection to establish the connection to the cloudlet or the cloud. The Cloudlet platform used in these experiments runs on a laptop Toshiba L755-1DR, Table 4 presents the main characteristics of this equipment [88]. On a remote AWS server run the cloud, as explained in the subsection before.

Table 4 - Cloudlet specifications

Toshiba SATELLITE L755-1DR specifications	
Processor	Intel® Core™ i7 -2670QM - clock speed : 2.20 / 3.10 Turbo GHz
OS	Ubuntu 14.05
Memory	RAM size: 8.0 GB ddr3, ROM size: 640.0 GB
Connectivity	Wi-Fi 802.11 a/b/g/n 2.4+5GHz, Bluetooth 4.0
Physical specification	Dimension (mm) 380.0 x 250.0 x 27.7, Weight (Kg) 2.5

5.3. USE CASE 1 - FLUID MOBILE APPLICATION

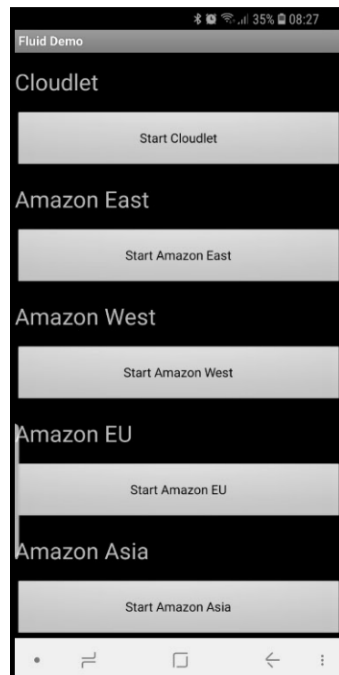
The first application is Fluid, an application used in interactive computer graphics representative of real-time games.

5.3.1. FLUID

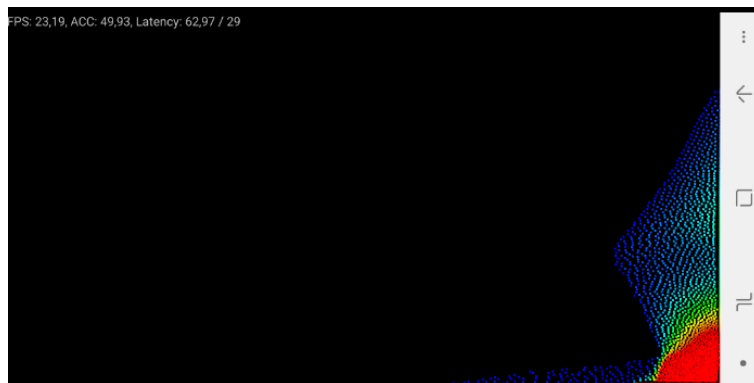
Fluid is a simple implementation of a liquid fluid simulation using the Smoothed Particle Hydrodynamics (SPH) method [89]. The SPH model is a Lagrangian method used to model fluid flow by treating each particle as a discrete element of fluid. It turns the device screen into a container where a liquid sloshes through the movement given into the device and the accelerometer input readings. It allows a user to interact directly with each particle.

The application backend runs on an Ubuntu OS and performs the dynamic simulation using 2218 particles. These particles slosh each side of the smartphone screen with different speed and direction through direct user interaction. On the backend side, the application runs a

physic simulator based on the predictive-corrective incompressible smoothed particle hydrodynamics [89].



(a) Cloudlet and Clouds server lists



(b) Application running

Figure 35 - Fluid client application

5.3.2. CLIENT APPLICATION

The Fluid client application is an Android application that runs on smartphones. Fluid configuration first step consists of choosing a server from a list and requesting the creation

of client threads. Figure 35-a) shows a list of servers installed on a Cloudlet and on an AWS Cloud.

After that, the particles will move around according to the speed and acceleration sensed by the smartphone, as presented in Figure 35 b). This client application sends the realtime readings to a graphics engine in the backend server; those readings are subject to a physics-based simulation and are periodically rendered on the smartphone, giving the illusion of liquid sloshing around.

The client application was configured to show the key features in the left top side of the screen, while the simulation is running. In that way, it provides the latency value and the output frame rate, as shown in Figure 36.

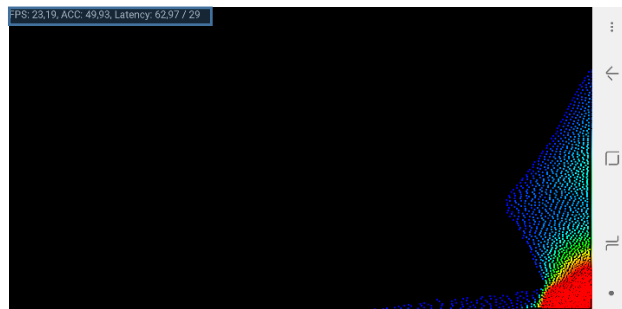


Figure 36 - Fluid client application running

5.3.3. SERVER

The cloudlet server is the main element of the architecture, it implements the compute-intensive backend, offloaded from the smartphone to the cloudlet or the Amazon EC2 cloud. The device movement generated by the smartphone user results on an accelerometer sensing and the communication of the readings to a graphics engine in the backend.

Figure 37 shows the TCP communication between the client application and the server. Figure 37 (a) represents the communication between client and server, we can observe that the client streams the information gathered by the accelerometer, the request part is only a few bytes. In the other way, the response data is the state of the simulated form and it has a much larger size (1400 B), as shown in Figure 37 (b).

No.	Time	Source	Destination	Protocol	Length	Info
7043	5.314000	192.168.0.10	192.168.1.234	TCP	54	32763 → 9093 [ACK] Seq=2769 Ack=3925865 Win=4096 Len=0
7044	5.314000	192.168.1.234	192.168.0.10	TCP	1454	9093 → 32763 [PSH, ACK] Seq=3925865 Ack=2769 Win=4096 Len=1400
7045	5.314000	192.168.0.10	192.168.1.234	TCP	54	32763 → 9093 [ACK] Seq=2769 Ack=3927265 Win=4096 Len=0
7046	5.314000	192.168.1.234	192.168.0.10	TCP	150	9093 → 32763 [PSH, ACK] Seq=3927265 Ack=2769 Win=4096 Len=96
7047	5.315000	192.168.0.10	192.168.1.234	TCP	54	32763 → 9093 [ACK] Seq=2769 Ack=3927361 Win=4096 Len=0
7048	5.315000	192.168.1.234	192.168.0.10	TCP	1454	9093 → 32763 [PSH, ACK] Seq=3927361 Ack=2769 Win=4096 Len=1400
7049	5.315000	192.168.0.10	192.168.1.234	TCP	54	32763 → 9093 [ACK] Seq=2769 Ack=3928761 Win=4096 Len=0
7050	5.315000	192.168.1.234	192.168.0.10	TCP	1454	9093 → 32763 [PSH, ACK] Seq=3928761 Ack=2769 Win=4096 Len=1400
7051	5.315000	192.168.0.10	192.168.1.234	TCP	54	32763 → 9093 [ACK] Seq=2769 Ack=3930161 Win=4096 Len=0
7052	5.315000	192.168.1.234	192.168.0.10	TCP	1454	9093 → 32763 [PSH, ACK] Seq=3930161 Ack=2769 Win=4096 Len=1400
7053	5.315000	192.168.0.10	192.168.1.234	TCP	54	32763 → 9093 [ACK] Seq=2769 Ack=3931561 Win=4096 Len=0

(a) Communication between client and server

> Frame 7054: 1454 bytes on wire (11632 bits), 1454 bytes captured (11632 bits)	
> Ethernet II, Src: Cisco_c0:a8:01 (00:00:0c:c0:a8:01), Dst: Htc_01:02:03 (00:09:2d:01:02:03)	
> Internet Protocol Version 4, Src: 192.168.1.234, Dst: 192.168.0.10	
▼ Transmission Control Protocol, Src Port: 9093, Dst Port: 32763, Seq: 3931561, Ack: 2769, Len: 1400	
Source Port: 9093	
Destination Port: 32763	
[Stream index: 0]	
[TCP Segment Len: 1400]	
Sequence number: 3931561 (relative sequence number)	
[Next sequence number: 3932961 (relative sequence number)]	
Acknowledgment number: 2769 (relative ack number)	
0101 = Header Length: 20 bytes (5)	
> Flags: 0x018 (PSH, ACK)	
Window size value: 4096	
[Calculated window size: 4096]	
[Window size scaling factor: -2 (no window scaling used)]	
Checksum: 0xde79 [unverified]	
[Checksum Status: Unverified]	
Urgent pointer: 0	
> [SEQ/ACK analysis]	
TCP payload (1400 bytes)	
▼ Data (1400 bytes)	
Data: 099a7b3fc34ec3becd67083f98b625bf0093453f21cb4cbf...	
[Length: 1400]	
0030	10 00 de 79 00 00 09 9a 7b 3f c3 4e c3 be cd 67 ...y... {?.N...g
0040	08 3f 98 b6 25 bf 00 93 45 3f 21 cb 4c bf 8c 4a ...?..%... E?!..L..J
0050	5d 3f bb ed 28 bf cc 37 75 3f 2f 67 cc be cf 03]?..(..7 u?/g....
0060	6f 3e 67 ca 1c bf f0 e0 77 3f ad fc 0d bf c7 ab o>g..... w?.....
0070	9f 3e 83 7e f3 be fe 07 e9 bc a9 b7 3c bf 5c 7d ..>~..... <...<.\}
0080	65 3f 7b 9c 67 bf d3 53 6b 3f 39 81 b1 bd de 9b e?{.g..S k?9.....
0090	25 3f 83 b7 24 bf 7d 9e 6d 3f 72 0e 64 bf d8 6f %?...\$.}. m?r.d..o
00a0	2f 3f 61 06 08 bf fa 3e d1 3e f0 13 3e bf 79 ff /?a....> ..>..y.
00b0	7f 3f 7f 5b 5f be e3 4f 78 3f 5e 6b f6 be bf 2d ..?[_...o x?^k...-
00c0	7b 3f 92 51 38 bf 67 b1 7a 3f 55 3f e2 be 92 ff {?.Q8.g. z?U?....
00d0	7f 3f 0f 4a 6e bf 87 66 20 3f b3 c8 3c bf 71 f3 ..?.Jn...f ?..<.q.
00e0	4d 3f 58 14 32 bf 1c 9b 71 3f 9b c2 2a bf d2 7d M?X.2... q?..*..}
00f0	70 3f a3 71 55 bf a2 41 7b 3f 39 e6 7b bf 2c 54 p?.qU..A {?9.{.,T

(b) Frame composition

Figure 37 - Communication sniffing

5.3.4. TEST SETUP

The first experiment consisted on allowing the dynamic resource allocation on the cloudlets. Cloud previously stores the VM images, but it does not present fast options to instantiate new images, resulting in long waiting time over WAN networks. For that reason, Cloudlets are expected to be much more agile in provisioning fast and dynamic solutions.

An user that needs to use a specific application shall connect to a nearby Cloudlet which must provide fast VM instantiation through the server backend application. If nearby cloudlets exist, the rapid provisioning enables the user a good quality of service at any place and

any time; and a precise VM image loads the application offloading components. Therefore, if the Base VM already instantiated on the cloudlet, it will lack only the application and library difference, also called the VM overlay.

To achieve fast provisioning, the Base VM and the VM overlay are two tasks done offline and register the time of the VM synthesis.

The base VM launched has a freshly installed OS Ubuntu 14.05, with 8 GB of disk and 1 GB of memory. An image was instantiated and after that the application and the other binaries were also configured. After that, the VM was ready to launch and create the overlay using the difference between the launch VM and the Base VM image.

The measurement of the VM synthesis time is illustrated in the Figure 38, so after the download of the VM overlay through Wi-Fi, decompress and instantiate on the Base VM and it stops when the construct launch booted VM. The measurement was made 10 times to collect the results during that period.

The second part of the experiment regards the quality of experience and interactivity to ensure that the delay between the input and the result output should be around 100 ms. In this scenario, the simulator runs all the 2218 particles with time steps of 20 ms, so it can generate up to 50 frames per second.

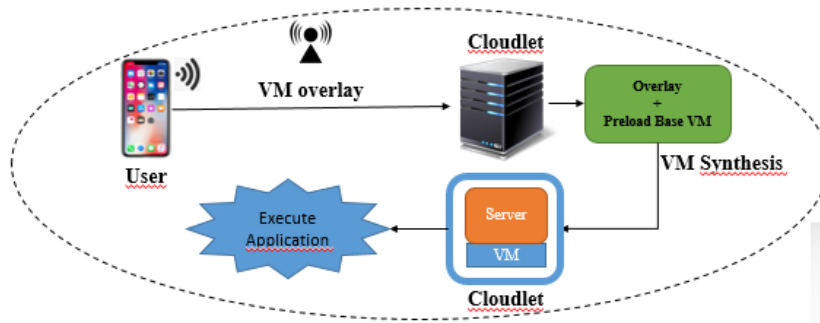


Figure 38 - VM Synthesis process

The performance evaluation measures the response time of the system using the cloudlet in comparison with the cloud through the workbench (Fig. 34) explained in subsection 5.2 .

The measurement starts from the moment the device sends the sensor data; then it processes the data until the server sends back the estimated prevision feedback. One measuring cycle

provides the response time given by several iterations performed by the user. The measurement process starts with a 5 minutes of interaction, generating almost 300 samples. During that time, the user has to slosh the particles quickly from one side to the other side of the smartphone screen. At the end of each cycle the measurements are collected.

The graphics quality determines the end-to-end latency between the sensing and display in the front-end side and the simulation on the back-end side. The quality of the graphics degrades and presents sudden or slow movements because of the latency increase. To get a fluid movement of the particles, the latency should present a maximum value of 100 ms, over wise jerky or sluggish can appear, deteriorating the user experience.

The other key feature is the output frame rate measured in frames per second (FPS), this value is a good metric to verify the graphics quality. This value can be compared with the value generated by the server. The server keeps generating up to 50 FPS and changes the states of simulation according to the data received from the sensors.

The round of the experiments performed with different VM instances on both cloudlet and cloud also allowed to analyse the influence of the server processing capacity in the system performance. So, cloud servers were instantiated with 1, 2 and 4 cores both in cloudlet and cloud and all the test cycles were performed, analysed the collected data and assess the performance of the implemented cloudlet prototype against the cloud.

5.4. USE CASE 2 – FACESWAP MOBILE APPLICATION

The second application is FaceSwap, an application used in face recognition representative of real-time face recognition. We use it to implement a use case scenario to visualize differences cloudlet can make in reducing network latency for compute-intensive and latency-sensitive applications.

5.4.1. FACESWAP

FaceSwap is an Android application that swaps people's faces in real time. This application used other applications to perform the swap, such as face tracking, face detection, and face recognition. The server setup also needs the installation of the dependencies OpenCV, OpenFace and Gabriel.

5.4.2. CLIENT APPLICATION

The FaceSwap Android Client can be download on Google Play platform [90], this application is really user-friendly with a minimum of function or configuration, so it is easy to get quick results from the principal screen presented in Figure 38.

The first part regards face tracking and detection, and it requires configuring the application. The first step is to add FaceSwap Server IPs, to access this menu it is necessary to select the option “Manage servers” inside the Menu Button on the top right corner of the principal page. Figure 39 presents the menu used to configure and save a cloud or cloudlet server with key features: name, IP address, server’s category: cloud or cloudlet.

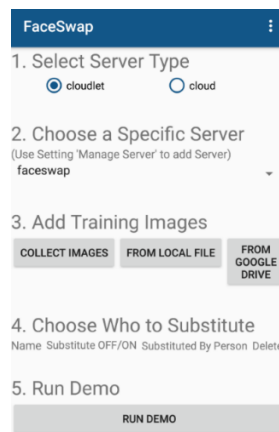


Figure 39 - FaceSwap android application

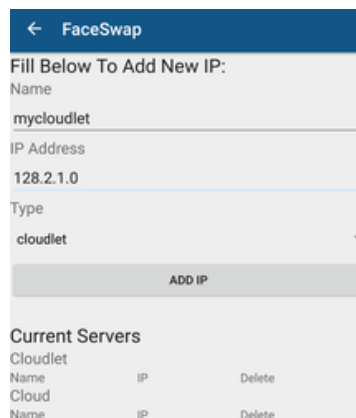
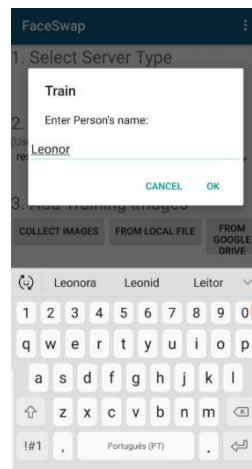


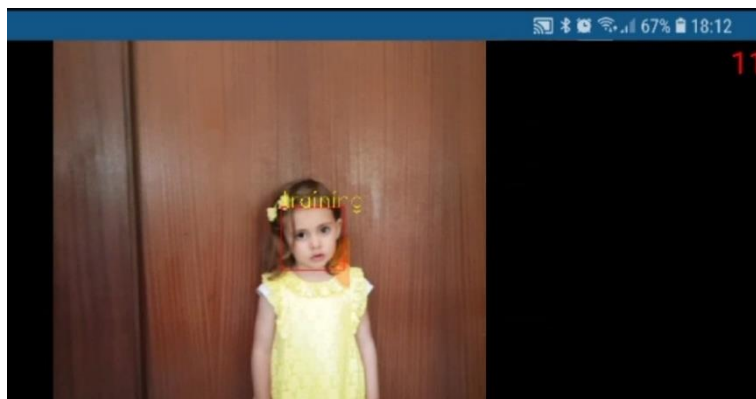
Figure 40 - Server configuration

The second step is to come back to the principal screen, and the user has to select the specific server in the top section “Select server type” from the 2 options cloud or cloudlet.

The third step is to start the training. The user has three different methods in the menu “Add Trainings Images”. If the user chooses the option “Collect images” it will automatically open the smartphone camera to collect training images. The option “From Local File” will allow the user to load a FaceSwap dataset stored on a local directory in the smartphone. The last option “From Google Drive” allows the user to load a FaceSwap dataset stored on Google Drive. In this part, the user has to enter the name of the person that will perform the experiment, as shown in Figure 40 a).



(a) Training Person Name personalization

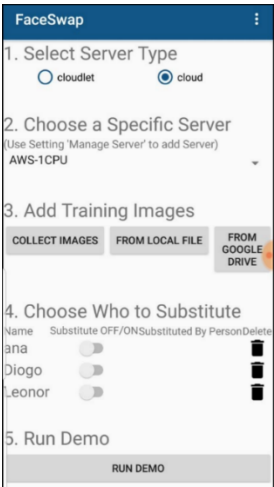


(b) Training images collect

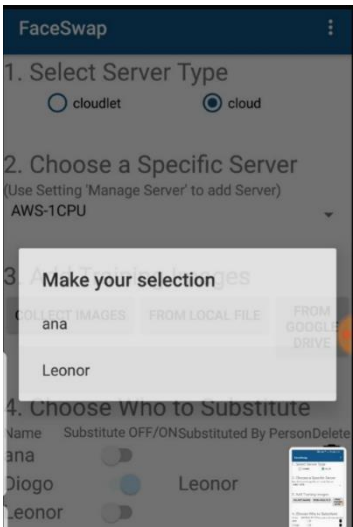
Figure 41 - FaceSwap Training session

After that, it will start the collection of the images. The person should turn the head to all directions to take pictures of many face profiles as possible. These different snapshots will turn the detection and recognition easier and faster.

“Choose faces to substitute” is the fourth step. The user can choose the person who will have the face swap with a substituted image. In that section, a list of persons that perform the training appears as shown in Figure 41 a). The user has to select the person who will have the face swap with another one. The user will choose that person from a list of persons from a new window as presented in Figure 41 b).



(a) List of Persons trained



(b) Swap Person Selection

Figure 42 - FaceSwap choose option menu

The experimental part starts from this point. The user has to select the option “Run demo” and it begins to stream images from the smartphone to the back-end server.

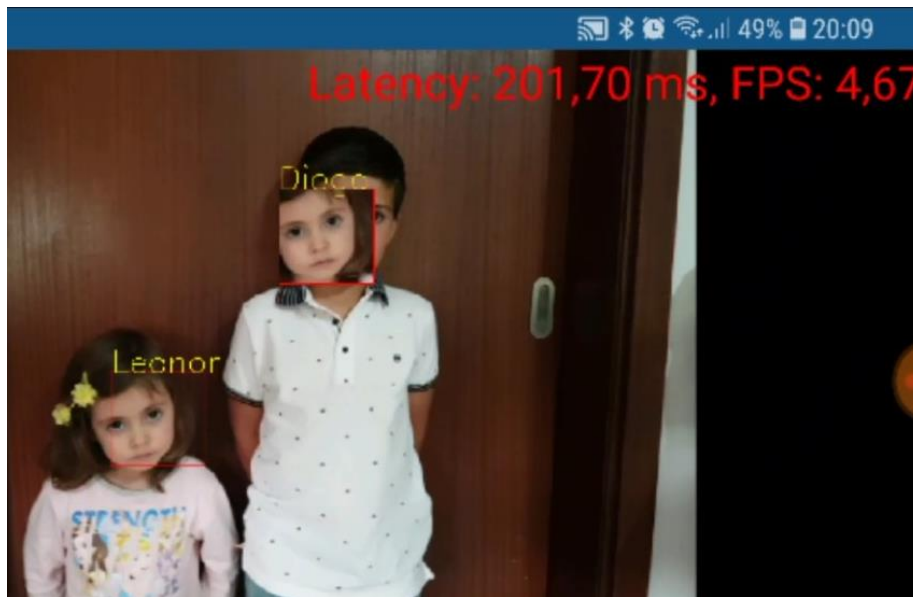


Figure 43 - Swap person selection

The key features of this experiment appear on the screen in real time. Figure 43 presents this feature on the right top of the screen where it appears the latency value and the output frame.

5.4.3. SERVER

The back-end server uses an Apache 2.0 license and a three-tier hierarchy based on face tracking, face detection, and face recognition, as presented in Figure 43.

The application client is constantly transmitting images of the faces in JPEG format, with dimension 640x480. On the back-end server, if the result of a face tracking is positive, it will send bounding boxes with faces in JPEG formats. If it establishes face detection and recognition, it will perform the swap of the person's faces. The communication between the Android client application and the server uses TCP traffic on the ports 9098 and 9101.

The FaceSwap server setup installs dependencies, such as dlib, OpenCV, OpenFace, and Gabriel.

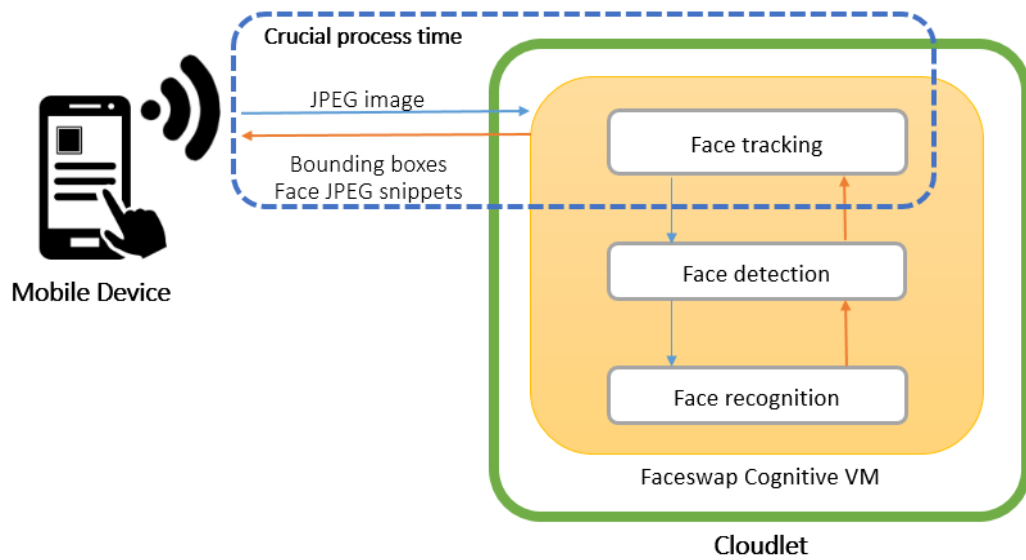


Figure 44 - Application client server process

Gabriel is the dependency responsible for the underlying communication library [91]. Gabriel is a wearable cognitive assistant for users in cognitive decline. It combines the image capture and sensing capabilities of Google Glass devices with cloudlet processing to perform real-time scene interpretation. This system layers on top of an OpenStack extension for cloudlet environments.

```

_main_      INFO      Start RESTful API Server (port :8021)
_main_      INFO      Start UPnP Server
_main_      INFO      Start monitoring offload engines
gabriel.control.mobile_server INFO * Mobile server(<class 'gabriel.control.mobile_server.MobileVideoHandler'>) configuration
gabriel.control.mobile_server INFO - Open TCP Server at ('0.0.0.0', 9098)
gabriel.control.mobile_server INFO - Disable nagle (No TCP delay) : 1
gabriel.control.mobile_server INFO -----
gabriel.control.mobile_server INFO * Mobile server(<class 'gabriel.control.mobile_server.MobileAccHandler'>) configuration
gabriel.control.mobile_server INFO - Open TCP Server at ('0.0.0.0', 9099)
gabriel.control.mobile_server INFO - Disable nagle (No TCP delay) : 1
gabriel.control.mobile_server INFO -----
gabriel.control.mobile_server INFO * Mobile server(<class 'gabriel.control.mobile_server.MobileResultHandler'>) configuration
gabriel.control.mobile_server INFO - Open TCP Server at ('0.0.0.0', 9101)
gabriel.control.mobile_server INFO - Disable nagle (No TCP delay) : 1
gabriel.control.mobile_server INFO -----
gabriel.control.app_server INFO * Application server(<class 'gabriel.control.app_server.VideoSensorHandler'>) configuration
gabriel.control.app_server INFO - Open TCP Server at ('0.0.0.0', 10101)
gabriel.control.app_server INFO - Disable nagle (No TCP delay) : 1
gabriel.control.app_server INFO -----
gabriel.control.app_server INFO * Application server(<class 'gabriel.control.app_server.AccSensorHandler'>) configuration
gabriel.control.app_server INFO - Open TCP Server at ('0.0.0.0', 10102)
gabriel.control.app_server INFO - Disable nagle (No TCP delay) : 1
gabriel.control.app_server INFO -----
gabriel.control.ucomm_relay INFO * UComm server configuration
gabriel.control.ucomm_relay INFO - Open TCP Server at ('0.0.0.0', 9090)
gabriel.control.ucomm_relay INFO - Disable nagle (No TCP delay) : 1
gabriel.control.ucomm_relay INFO -----
gabriel.common.upnp_client INFO execute : java -jar /usr/local/bin/gabriel_upnp_client.jar
gabriel.common.upnp_client INFO User communication module is connected
gabriel.common.upnp_client INFO execute : java -jar /usr/local/bin/gabriel_upnp_client.jar
gabriel.proxy.common INFO Gabriel Server :
gabriel.proxy.common INFO { 'u'acc_tcp_streaming_address': u'10.2.4.17',
gabriel.proxy.common INFO   'u'acc_tcp_streaming_port': 10102,
gabriel.proxy.common INFO   'u'result_return_server_list': [u'10.2.4.17:10120'],
gabriel.proxy.common INFO   'u'ucomm_data_address': u'10.2.4.17',
gabriel.proxy.common INFO   'u'ucomm_data_port': 9090,
gabriel.proxy.common INFO   'u'video_tcp_streaming_address': u'10.2.4.17',
gabriel.proxy.common INFO   'u'video_tcp_streaming_port': 10101 }
gabriel.proxy.common INFO Success to connect to (u'10.2.4.17', 10101)
gabriel.control.app_server INFO Offloading engine is connected
gabriel.proxy.common INFO Start getting data from the server
gabriel.proxy.common INFO Start publishing data
gabriel.control.mobile_server INFO Google Glass is connected for (Mobile Result Handler)
gabriel.control.mobile_server INFO Google Glass is connected for (Mobile Video Server)

```

Figure 45 - Server launching control

Gabriel runs the FaceSwap engine on the VM instance, at first it uses Universal Plug and Play (UPnP) to discover control server from the ucomm server and cognitive engines, as shown in Figure 45. If the ucomm server connects successfully to the control server, a log message “INFO User communication module is connected” appears at the control server. After that, it runs the cognitive engine that will prompt the message “INFO offloading engine is connected”. Figure 46 a-) present the communication between the user application and the server during a face detection. The user has to take several pictures to form a little database of faces in variable angles, so the detection phase will be easier to identify the person.

```

gabriel.control.mobile_server INFO Mobile Result Handler terminate thread
gabriel.control.mobile_server DEBUG Recv 0 data at Mobile Video Server

gabriel.control.mobile_server INFO Mobile Video Server terminate thread
gabriel.control.mobile_server INFO Google Glass is connected for (Mobile Result Handler)
gabriel.control.mobile_server INFO Google Glass is connected for (Mobile Video Server)
gabriel.proxy.common INFO New connection is starting at 1526836123.294921
face_swap INFO training: sucesss - detected 1 face. add frame
gabriel.proxy.common INFO returning result: {"training": "Diogo", "engine_id": "dumm
gabriel.control.mobile_server INFO result message ({"training": "Diogo") sent to the Glass
face_swap INFO training: sucesss - detected 1 face. add frame

```

(a) Training for face detection

```

gabriel.proxy.common INFO      returning result: {"result": "\\type\\: \\detect\\", \\val
gabriel.control.mobile_server INFO      result message ({\"id\": 287, \"sensor_}) sent to the Glass
face_swap DEBUG      all image caught up! # images 7
face_swap DEBUG      bg-thread getting detection updates
face_swap DEBUG      cleared recognition busy
face_swap DEBUG      server response: {\"type\": \"FRAME_RESP\", \"name\": \"Diogo\",
face_swap DEBUG      image width: 640
face_swap DEBUG      # faces tracking 2
face_swap DEBUG      server busy event not set
face_swap DEBUG      server response: {\"type\": \"FRAME_RESP\", \"name\": \"Leonor\",
face_swap DEBUG      tracker run: 2.91204452515
face_swap DEBUG      tracker run: 2.65908241272
face_swap DEBUG      # faces in image: 2
face_swap DEBUG      bg-thread updating faces from detection process!
face_swap DEBUG      main process received recognition resp (\"type\": \"FRAME_RESP\", \"name\": \"Diogo\", \"success\": true, \"id\": 6)
gabriel.proxy.common INFO      returning result: {"result": "\\type\\: \\detect\\", \\val
face_swap DEBUG      main process received recognition name Diogo
face_swap DEBUG      main process received recognition resp (\"type\": \"FRAME_RESP\", \"name\": \"Leonor\", \"success\": true, \"id\": 7)
face_swap DEBUG      main process received recognition name Leonor
gabriel.control.mobile_server INFO      result message ({\"id\": 290, \"sensor_}) sent to the Glass
face_swap DEBUG      image width: 640
face_swap DEBUG      # faces tracking 2
face_swap DEBUG      tracker run: 1.80578231812
face_swap DEBUG      tracker run: 1.7409324646
face_swap DEBUG      # faces in image: 2

```

(b) Face swap during faces recognition

Figure 46 - Cognitive engine communications

OpenFace is the dependency used to perform both the faces classifier training and face recognition [92].

OpenFace's core provides a feature extraction method to collect a low-dimensional representation of any face. This function creates a face classifier by using a deep neural network (DNN) model to train and use a classification model.

The comparison feature outputs the similarity between two faces, and two faces are more likely the same person if it presents a lower score.

The face recognition application detects faces in an image and attempts to identify the face from a pre-populated database. The Haar Cascade of classifiers collects the detection part and the Eigenfaces method allows the identification based on the principal component analysis (PCA) [93]. OpenCV implements image processing and computer vision routines [94].

Our experiments only consider the recognition part on a trained system because the classifiers train and the database population are jobs done offline. The Figure 46 b) presents the engine communications during the swap between two faces. If the faces detected match the faces present in the database, then a successful communication reply is sent out to enable the swap of the faces.

The FaceSwap server accomplishes the setup in two manners: manually or using a pre-packaged image. In the first approach, it is necessary to install all components and their dependencies: OpenFace, dlib, OpenCV, Torch, Gabriel and at last FaceSwap. We tried several times. It comes to several errors regarding the different versions of pip or the docker installation for OpenFace.

We assume that it was preferable to use the pre-packaged image, in the qcow2 format as shown in Figure 47, and OpenStack imports it as a volume. After that, it launches an instance directly, and the Faceswap server already launches by itself on the start-up. The inconvenience is that we assume that the possibility to not make an overlay because the image comes already with the Faceswap server on it. Another interesting part is that the Faceswap server is already part of the AWS EC2 VM instance, so it is possible to launch with a different configuration regarding the number of cores and RAM width.

Property	Value
Property 'description'	Faceswap-Server No AVX
checksum	e09c4f0c69c80ba629494c746c3eb46c
container_format	bare
created_at	2018-01-30T22:11:52.000000
deleted	False
disk_format	qcow2
id	0625da53-db33-4030-85a7-2e32a790024d
is_public	True
min_disk	0
min_ram	0
name	Faceswap-Server2
owner	212445136b10495ea719d5846a29a653
protected	False
size	8208056320
status	active
updated_at	2018-01-30T22:17:33.000000

Figure 47 - FaceSwap image metadata

5.4.4. TESTBENCH SIMULATION SETUP

The main goal of the scenario implemented is to verify the crucial role that cloudlets play in reducing end-to-end latency for computation offloading mobile applications. This test used the FaceSwap application, that performs face tracking, detection, recognition and swap in real time.

Tests shown that face tracking performs quickly, around 15ms, and that it is possible collect data at high frame rates, making it possible to achieve a real-time response. On the other hand, as face detection and face recognition takes a longer times, around 200ms, these tasks are used offline, so the main goal is still attainable. When the results become available, trackers can be updated.

The performance evaluation measures the response time of the system using the cloudlet in comparison with the cloud through the workbench (Fig. 34) explained in subsection 5.2.

The measurement is regarding end-to-end latency, which is the time difference between the substituted face and the original face. One measuring cycle measures the response time given several iterations performed by a user. We start the process by launching the recommended FaceSwap server with 4 cores and 8 GB RAM on the cloudlet and on the cloud AWS EC2 Oregon. The measurement uses an N-cycle of interaction. We use children as users because they are always moving and we can analyze better features like face tracking. At the end of each cycle, we collect the desired measurements.

The end-to-end latency determines the quality of the graphics. To use this application fluidly, the latency should present a maximum value of 200 ms, however, the user experience will deteriorate.

The other key feature is the output frame rate measured in frame per second (FPS), this value is a good metric to verify the graphics quality and good value should be around 50 FPS.

The second round of experiments uses different VM instances on both the cloudlet and the cloud. It was analyzed the possibility of increasing the number of cores to achieve better results. Therefore, instances were launched with 2, 4, 6 and 7 cores on the cloudlet and with 1, 2, 4, 8, 16 and 32 cores on the cloud. The same simulation was performed and it was analyzed, we analyze the collected data and assess the performance of our implemented cloudlet prototype against the cloud.

6. ANALYSIS OF THE RESULTS

Experiments occurred in a fixed environment, deploying each scenario with the same experimental settings. From the comparison of cloud services, this work selects AWS EC2-West (Oregon) which presents a round trip time (RTT) of 101.5 ms, a better result than the ones measured on AWS EC2-Europe, AWS EC2-Asia, and AWS EC2-East. This value is higher compared to the values presented by Li et al. on his study, which reports an RTT of 74 ms from 260 global vantage points to their optimal Amazon EC2 instances.

The mobile device uses 802.11n to connect to a private access point connected to the network via Ethernet and then via the Internet to the AWS website portal. The mobile device uses Wi-Fi 802.11n to connect to the cloudlet is on the same Ethernet network as the access point.

The experiments evaluate end-to-end latency and user quality by implementing N-cycles of interaction that registers different phenomena that can occur in real life, such as bandwidth limitations, Wi-Fi saturation, and congestion, routing instability, application vendor failure. These N-cycles enable the result presentation and comparison with a reliable interval. For each use case, these results use Cumulative Distribution Function (CDF) that provides easily various information in one plot, such as the median, worst-case, best-case, standard deviation and also the percentile of response time or frame rate output in each simulation.

6.1. TEST RESULT USE CASE - FLUID

The aim is to verify the real-time value of the application's offloading part at the network edge servers. This use case uses an application representative of real-time games, which is one target for 5G requirements.

The first experiment tests and compares the efficiency of VM synthesis on the cloudlet against the cloud solution. It is important that it instantiates fast enough so that the user continues with a good quality of experience.

The second experiment compares the values of the response time and the frame rate output of similar VM instances running on both the cloudlet and the cloud. Table 5 summarizes this comparison. On the one hand, the AWS dashboard presents a list of instances with fixed configuration values, such as the number of cores or the RAM memory.

On the other hand, instances launched through the cloudlet present hardware limitations, so it is not possible to launch instances with the exact number of cores and RAM.

This experiment also analyzes the possibility of decreasing the values collected before by increasing the VM instances' size.

Table 5 - Servers instance configuration type

Servers	Instance configuration type		
	1 core	2 cores	4 cores
Cloudlet	1 CPU - 1 GB RAM	2 CPU - 4 GB RAM	4 CPU - 6 GB RAM
Cloud	1 CPU - 1 GB RAM	2 CPU - 4 GB RAM	4 CPU - 16 GB RAM

6.1.1. CLOUDLET MEASUREMENTS

The first experiment regards the VM synthesis time. It is possible to perform the VM synthesis after instantiating the Base VM and the VM overlay. Doing these offline tasks previously enables fast provisioning.

The Base VM launched is a new installation of an OS Ubuntu 14.05, with 8 GB of disk and 1 GB of memory. It gathers the VM overlay after instantiating the Base VM, installing and configuring the needed application and all necessary binaries. Figure 48 presents the overlay collected using the difference between the launch VM and the Base VM image.

Image Overview	
Information	
Name	overlay-167db325-aec5-41f2-872b-4abcf9fa05
ID	0e79bfaa-e533-4509-9126-4b40929975c8
Owner	212445136b10495ea719d5846a29a653
Status	Active
Public	Yes
Protected	No
Checksum	bec48993e019dd3a30877c8ae010d1aa
Created	Dec. 12, 2016, 10:33 p.m.
Updated	Dec. 12, 2016, 10:35 p.m.
Specs	
Size	2.4 MB
Container Format	BARE
Disk Format	RAW
Min Disk	8GB
Min RAM	1GB
Custom Properties	
Euca2ools state	available
Image Type	snapshot
base_image_ref	acf2c2b6-8f5d-4fe0-bb4e-be894f535430
base_resource_xml_str	<domain type="kvm"> <name>cloudlet-d71b7be5- unit="KiB">1048576</currentMemory> <vcpu plac abda52a61692094b3b7d45c9647d022f5e297d1 3bc774e1-92d1-4564-8563-7dbf9030e8d4 bb8fdccb-cb3b-4253-b01b-d77da933acd7 9c0e2e25-016b-4ca1-82f6-0386e747fee6 cloudlet_type cloudlet_overlay image_location snapshot instance_uuid 167db325-aec5-41f2-872b-4abcf9fa05 is_cloudlet True owner_id 212445136b10495ea719d5846a29a653 user_id 829d668992f945f69d2fb2e8036503d5

(a) VM overlay disk image file metadata

```
stack1@mickael-SATELLITE-L755:~$ glance --os-username=admin --os-password=nomoresecrete --os-tenant-name=demo --os-auth-url=http://localhost:5000/v2.0 image-show overlay-167db325-aec5-41f2-872b-4abcf9fa05
```

Property	Value
Property 'base_image_ref'	acf2c2b6-8f5d-4fe0-bb4e-be894f535430
Property 'base_resource_xml_str'	<domain type='kvm'> <name>cloudlet-d71b7be54b104dee82c9038fddbbf39c</name> <uuid>d71b7be5-4b10-4dee-82c9-038fddbbf39c</uuid> <memory unit='KiB'>1048576</memory> <currentMemory unit='KiB'>1048576</currentMemory> <vcpu placement='static'>4</vcpu> <os> abda52a61692094b3b7d45c9647d022f5e297d1b788679eb93735374007576b8 3bc774e1-92d1-4564-8563-7dbf9030e8d4 bb8fdbcbb-cb3b-4253-b01b-d77da933acd7 9c0e2e25-016b-4ca1-82f6-0386e747fee6 cloudlet_overlay snapshot available snapshot 167db325-aec5-41f2-872b-4abcf9fa05 True 212445136b10495ea719d5846a29a653 829d668992f945f69d2fb2e8036503d5 bec48993e019dd3a30877c8ae010d1aa bare 2016-12-12T22:33:34.000000 False raw 0e79bfaa-e533-4509-9126-4b40929975c8 True 8 1024 overlay-167db325-aec5-41f2-872b-4abcf9fa05 212445136b10495ea719d5846a29a653 False 2552680 active 2016-12-12T22:35:36.000000
Property 'base_sha256_uuid'	abda52a61692094b3b7d45c9647d022f5e297d1b788679eb93735374007576b8
Property 'cloudlet_base_disk_hash'	3bc774e1-92d1-4564-8563-7dbf9030e8d4
Property 'cloudlet_base_memory'	bb8fdbcbb-cb3b-4253-b01b-d77da933acd7
Property 'cloudlet_base_memory_hash'	9c0e2e25-016b-4ca1-82f6-0386e747fee6
Property 'cloudlet_type'	cloudlet_overlay
Property 'image_location'	snapshot
Property 'image_state'	available
Property 'image_type'	snapshot
Property 'instance_uuid'	167db325-aec5-41f2-872b-4abcf9fa05
Property 'is_cloudlet'	True
Property 'owner_id'	212445136b10495ea719d5846a29a653
Property 'user_id'	829d668992f945f69d2fb2e8036503d5
checksum	bec48993e019dd3a30877c8ae010d1aa
container_format	bare
created_at	2016-12-12T22:33:34.000000
deleted	False
disk_format	raw
id	0e79bfaa-e533-4509-9126-4b40929975c8
is_public	True
min_disk	8
min_ram	1024
name	overlay-167db325-aec5-41f2-872b-4abcf9fa05
owner	212445136b10495ea719d5846a29a653
protected	False
size	2552680
status	active
updated_at	2016-12-12T22:35:36.000000

(b) VM overlay memory snapshot metadata

Figure 48 - VM overlay metadata files

Installed on the mobile phone, the VM overlay has a size of 2,4 MB.

The measurement of the VM synthesis time follows Figure 38. So the VM overlay needs to be downloaded through Wi-Fi, decompress and instantiate on the Base VM and it stops when the constructed launch VM is booted. The test measurement occurs 10 times to collect different periods.

The VM synthesis presents an average of 29.84 seconds with a standard deviation of 0.45 seconds. These results are higher compared with the ones presented by Ha k. at el [95] on his study, it presents an average time under 10s. However, our time is under the synthesis time estimated as too large for good user experience between 60 and 150 seconds.

The second process launches the Fluid server on VM instances. After that, it is possible to launch Fluid client application on the mobile smartphone. During 5 minutes of interaction, which the user has to slosh the particles quickly from one side to the other side of the smartphone screen, we collect the desired measurements.

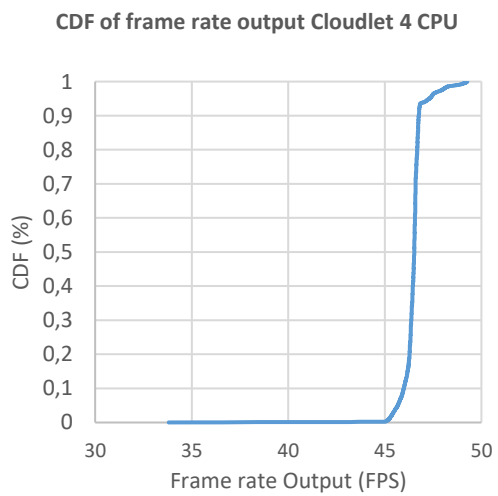
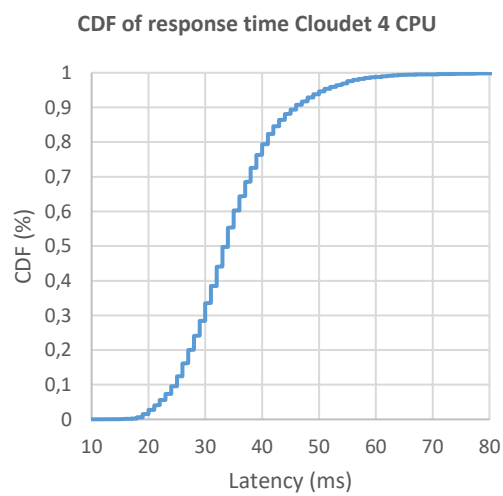
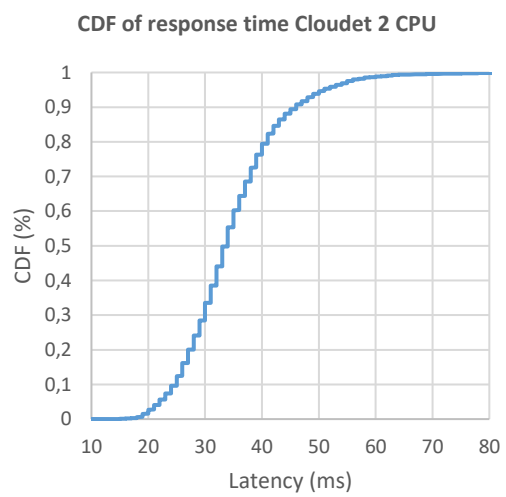
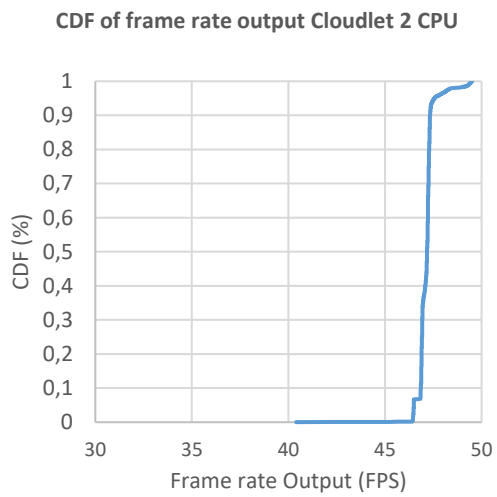
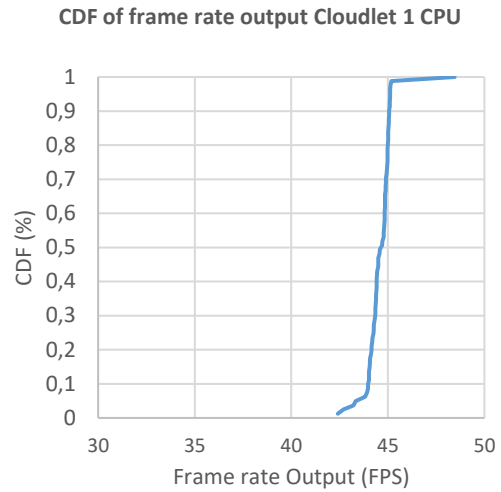
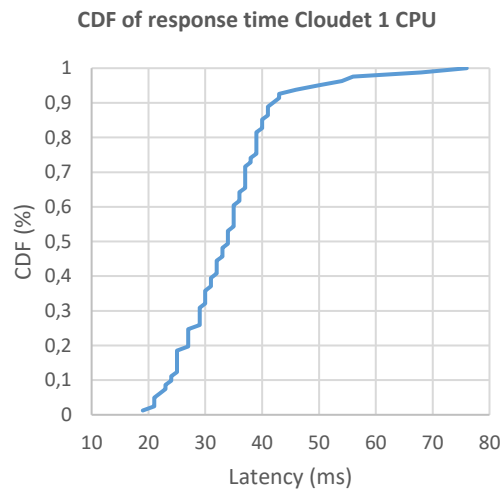


Figure 49 - CDF for response time and frame rate for Fluid - Cloudlet

Figure 49 and Table 6 present the test results of the Fluid application using Cloudlet's instances with 1, 2 and 4 CPU.

Table 6 - Cloudlet test results for Fluid

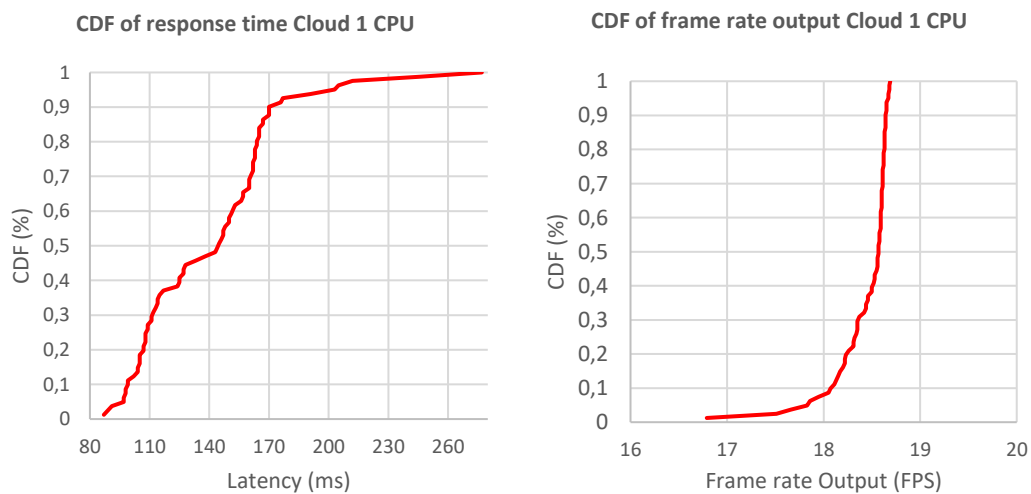
CLOUDLET	1 core		2 cores		4 cores	
	Latency	FPS	Latency	FPS	Latency	FPS
Median	34.0	44.7	34.0	47.2	32.0	46.5
Minimum	19.0	42.4	10.0	40.4	10.0	33.8
Maximum	76.0	48.5	91.0	49.5	68.0	49.3
Standard Deviance	9.4	0.7	8.9	0.4	7.5	0.6

6.1.2. CLOUD MEASUREMENTS

Figure 50 and Table 7 show the experiments using AWS EC2 Oregon and present the values of CDF regarding response time and frame rate output.

Table 7 - Cloud test results for Fluid

CLOUD	1 core		2 cores		4 cores	
	Latency	FPS	Latency	FPS	Latency	FPS
Median	145.0	8.6	126.0	22.0	124.0	21.7
Minimum	87.0	16.8	87.0	15.4	10.0	4.9
Maximum	277.0	18.7	328.0	23.8	605.0	22.6
Standard Deviance	35.6	0.3	34.1	0.9	38.6	2.1



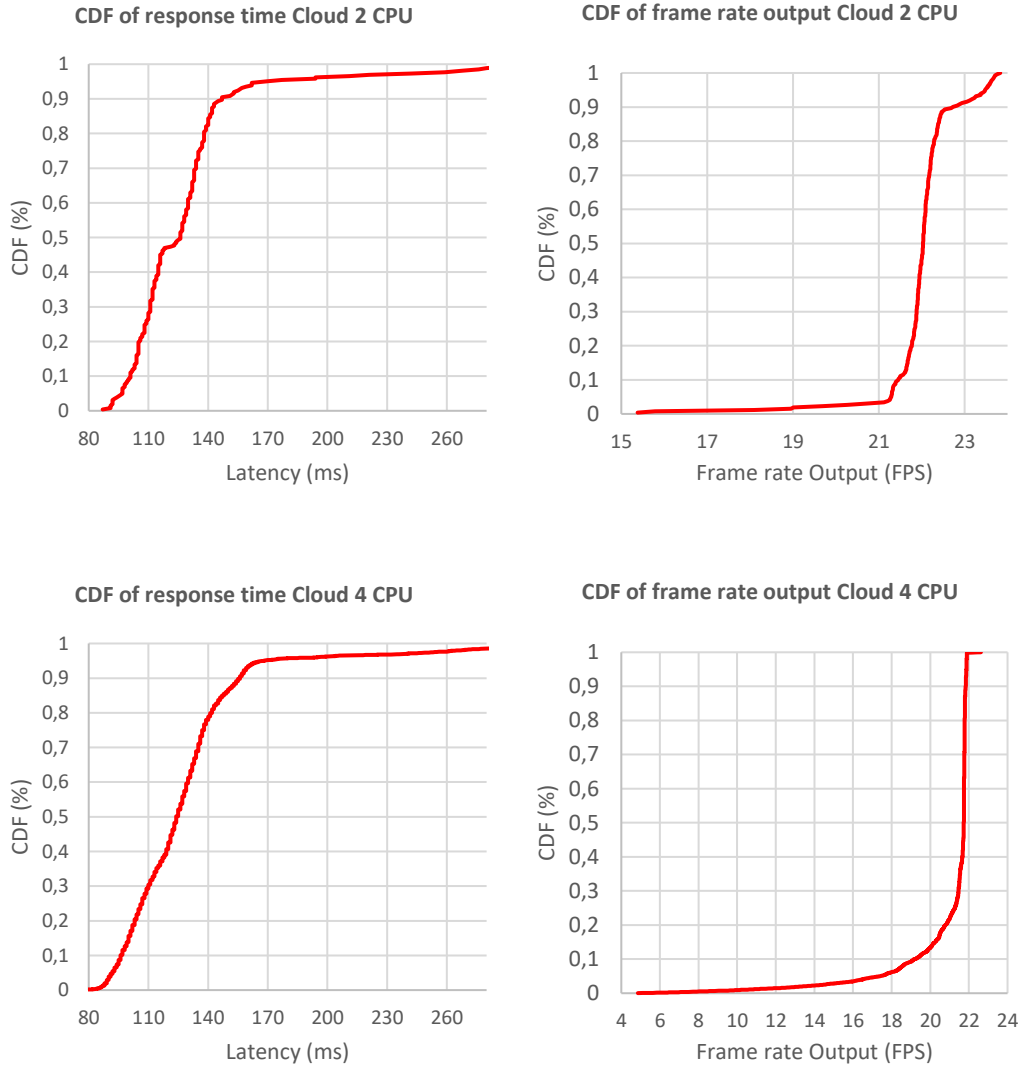


Figure 50 - CDF for response time and frame rate for Fluid - Cloud

6.1.3. RESULTS

Figure 51 presents the comparison between the Cloudlet and Cloud regarding the Fluid application's offloading on the servers.

The quality of the graphics is much higher in Cloudlet scenario, for Cloudlet response time median values are around 32-34 ms and for Cloud the median values are around 124-145 ms.

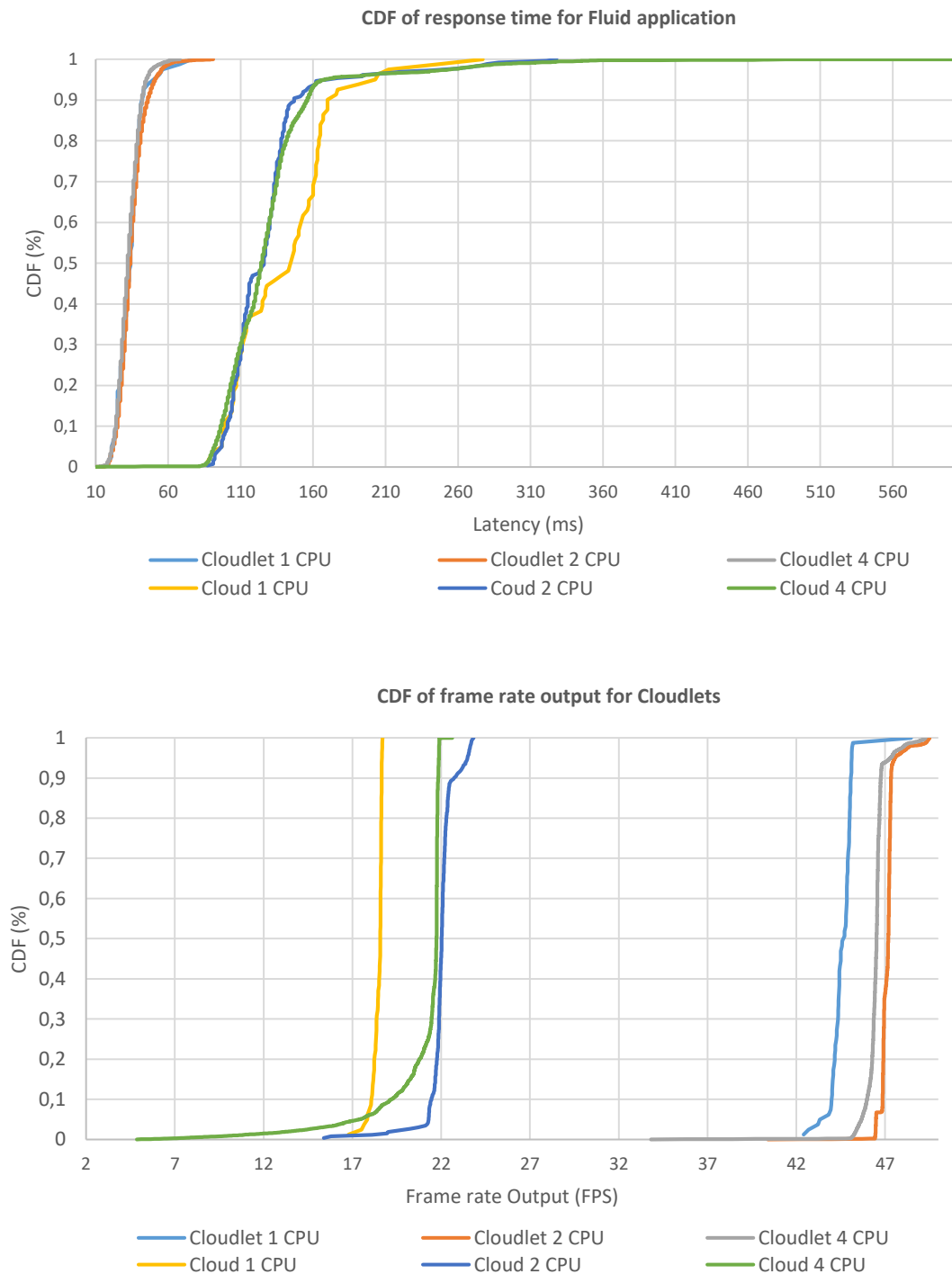


Figure 51 - Cloud and Cloudlet test result comparison for Fluid

Compared with Cloud AWS EC2 Oregon, Cloudlet present latency values almost lower four times. Using the cloudlet server, the movement of the particles is more fluid and does not degrade easily as on the cloud.

The other key feature analyzed is the output frame rate that present median values of 44-47 FPS for Cloudlet and 18-22 for Cloud. Using Cloudlet ensures a rate of at least two times higher than the cloud.

This 2 features analyzed to ensure that the use of Cloudlet can have a great impact and reduce brutally the latency and augment substantially the user quality of experience.

To use this application and verify a fluid movement of the particles, the latency should present a maximum value under 100 ms, over wise jerky or sluggish can appear, deteriorating the user experience [95].

The values presented by the Cloudlet successfully accomplish the application requirements. We also noticed that the values collected by K. Ha are similar compared to the results gathered in our experiments.

We also notice that the VM instances with a higher configuration of CPU and RAM can achieve better performance for intensive computation tasks in the server side. For Fluid application, and comparing the data from the VM instances with 1 CPU and 4 CPU, we verify that the latency reduces for 6% and an increase of 4% on the frame rate output.

6.2. TEST RESULT FROM USE CASE - FACESWAP

Our aim is to confirm the real value of offloading part of the application at the edge servers. We use the structure of this application representative of face recognition, which is one subject for 5G requirements.

The experiment allows us to compare directly the values of the response time and the frame rate output on our cloudlet and on AWS EC2-Oregon with similar VM instances, as summarized in Table 8. On one hand, AWS has instances with fixed values. On the other hand, instances launched in our cloudlet present hardware limitations, so it is not possible to launch instances with the exact number of cores and RAM.

This experiment also verifies the possibility of increasing the values collected before by increasing the VM instances' size and comparing many cases to achieve better performance.

Table 8 - Instance type configuration

Servers	Instance Type configuration						
	2 cores	4 cores	6 cores	7 cores	8 cores	16 cores	32 cores
Cloudlet	2 CPU 4 GB RAM	4 CPU 6 GB RAM	6 CPU 5 GB RAM	7 CPU 6 GB RAM			
Cloud	2 CPU 8 GB RAM	4 CPU 16 GB RAM			8 CPU 32 GB RAM	16 CPU 64 GB RAM	32 CPU 132 GB RAM

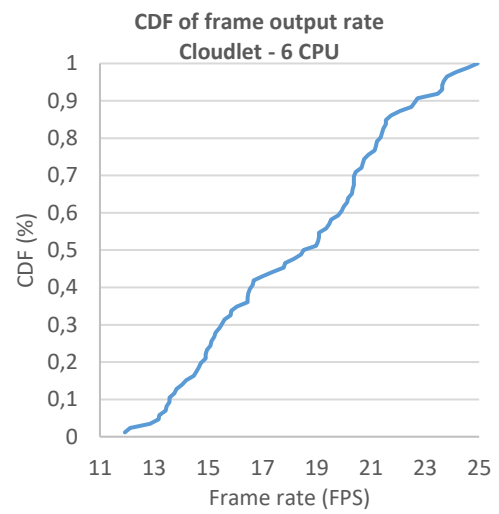
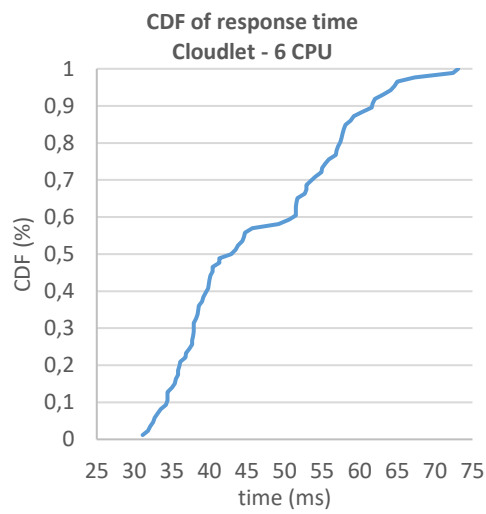
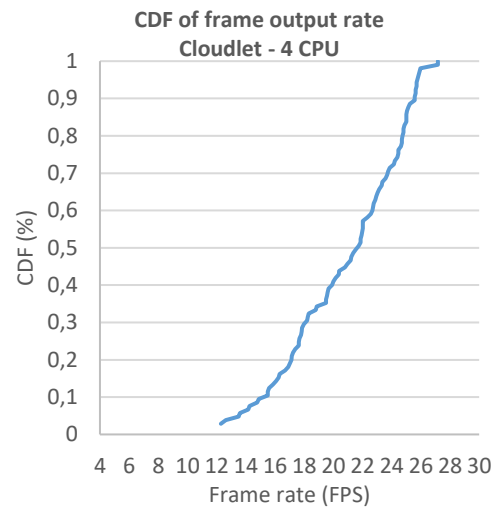
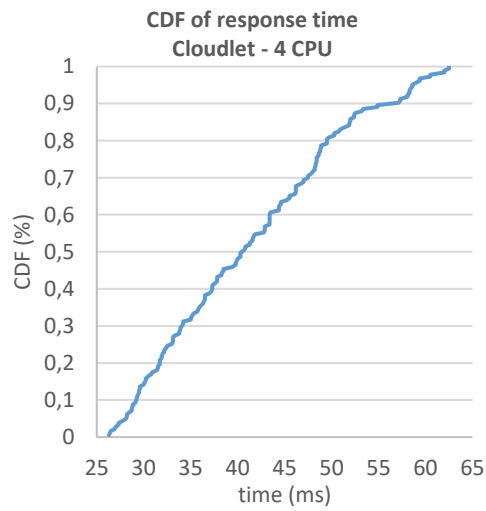
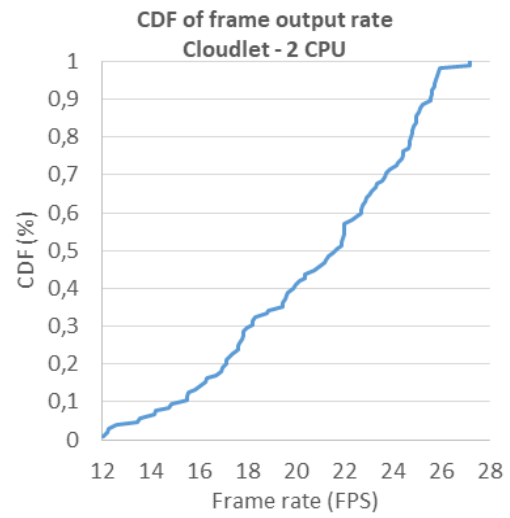
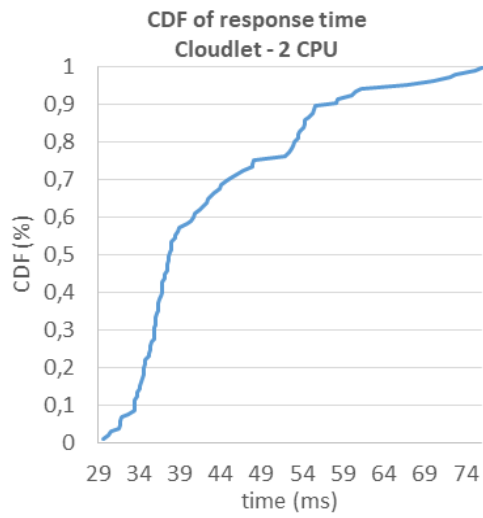
6.2.1. CLOUDLET MEASUREMENTS

The process launches FaceSwap server on VM instances after that it is possible to launch FaceSwap client application on the mobile smartphone and choose the cloudlet server. For the first Cloudlet instantiation, it is necessary to perform the faces training. After that, an option enables the user to choose two persons who will have to swap their faces for 100 images.

Figure 52 and Table 9 present the Fluid application's test results using the Cloudlet. We launch instances with 1, 2, and 4 CPU.

Table 9 - Cloudlet test results for FaceSwap

Cloudlet	2 CPU		4 CPU		6 CPU		7 CPU	
	Latency (ms)	FPS	Latency (ms)	FPS	Latency (ms)	FPS	Latency (ms)	FPS
Median	37.9	21.7	40.6	19.6	43.2	18.8	42.9	19.0
Minimum	29.7	12.1	26.3	4.9	31.1	11.9	34.0	12.0
Maximum	76.3	27.2	62.5	31.2	73.1	24.9	72.8	23.0
Deviance	10.8	3.9	9.7	5.3	10.9	3.4	9.7	3.0



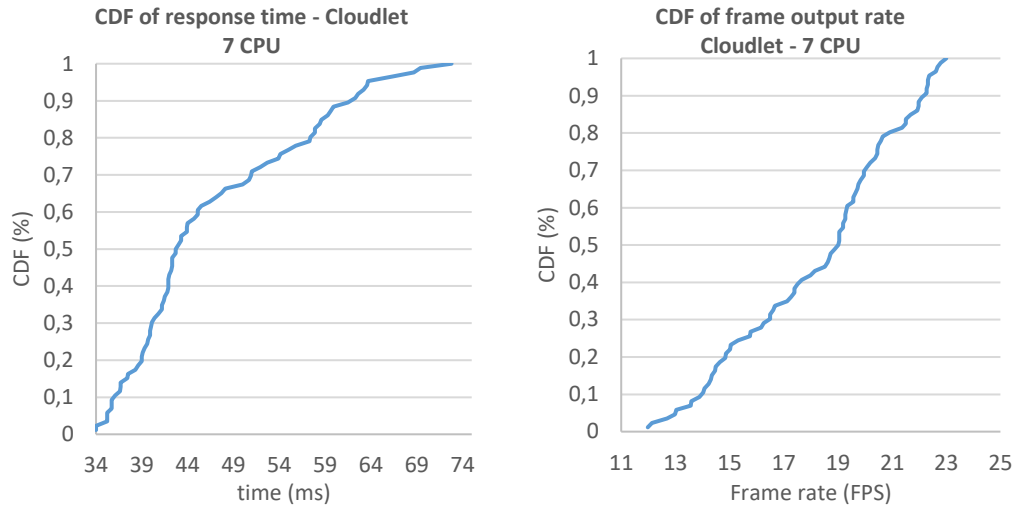


Figure 52 - CDF for response time and Frame rate for FaceSwap - Cloudlets

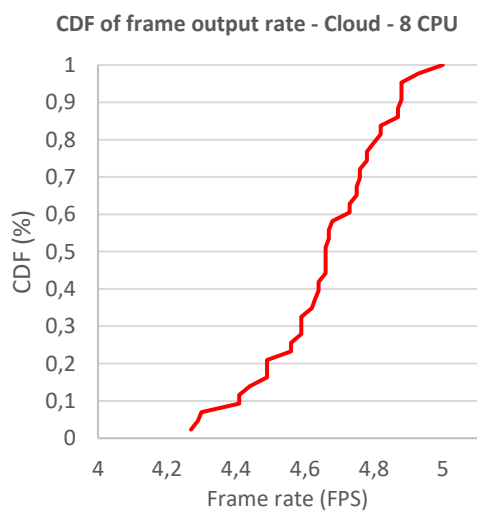
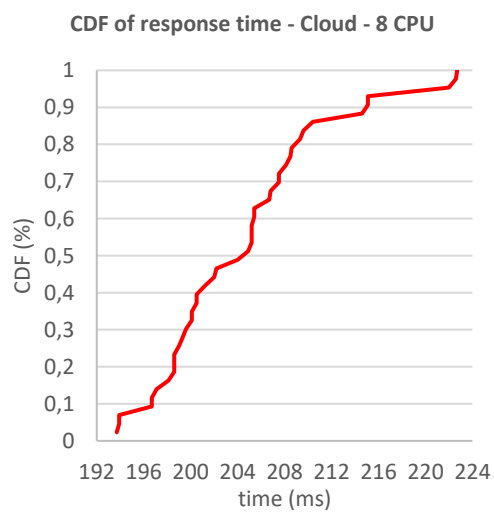
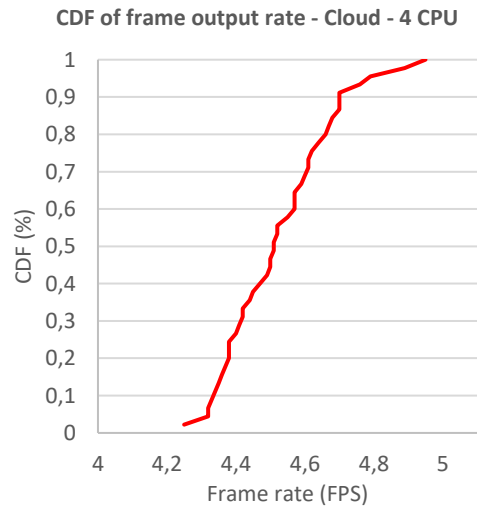
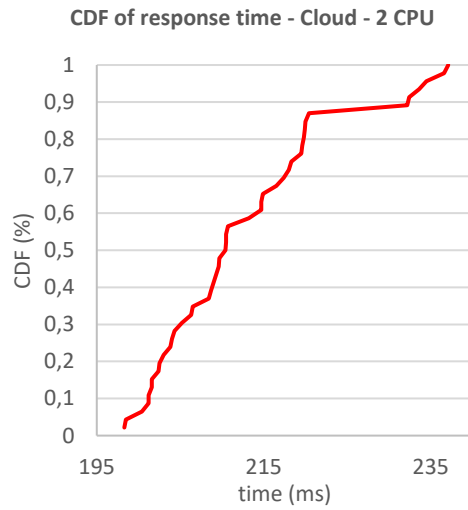
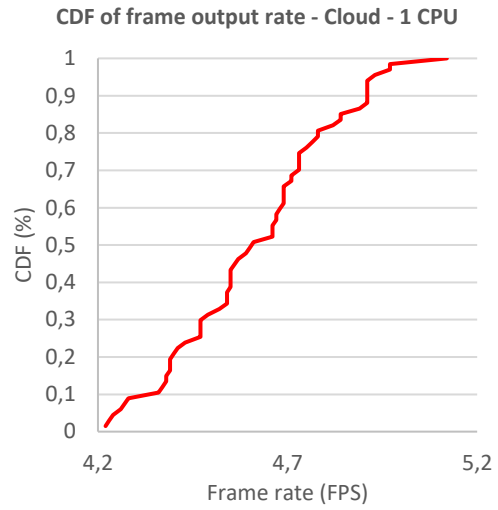
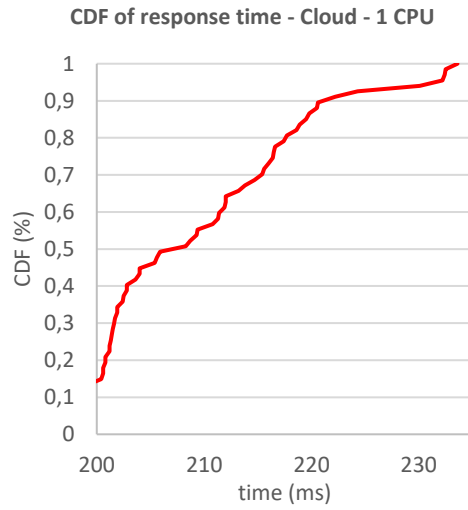
6.2.2. CLOUD MEASUREMENTS

Figure 53 and Table 10 present the values of CDF regarding response time and frame rate output using AWS EC2 Oregon.

Table 10 - Cloud test results for FaceSwap

Cloud	1 CPU		2 CPU		4 CPU	
	Latency (ms)	FPS	Latency (ms)	FPS	Latency (ms)	FPS
Median	208.3	4.6	210.5	4.6	212.4	4.5
Minimum	192.9	4.2	198.3	4.1	195.1	4.3
Maximum	233.6	5.1	237.1	5.0	251.9	5.0
Standard Deviance	10.3	0.2	10.5	0.2	8.9	0.2

Cloud	8 CPU		16 CPU		32 CPU	
	Latency (ms)	FPS	Latency (ms)	FPS	Latency (ms)	FPS
Median	204.9	4.7	618.5	1.6	379.9	2.6
Minimum	193.7	4.3	580.9	1.5	340.0	2.4
Maximum	222.7	5.0	676.5	1.7	412.0	2.9
Standard Deviance	7.3	0.2	27.9	0.1	17.2	0.1



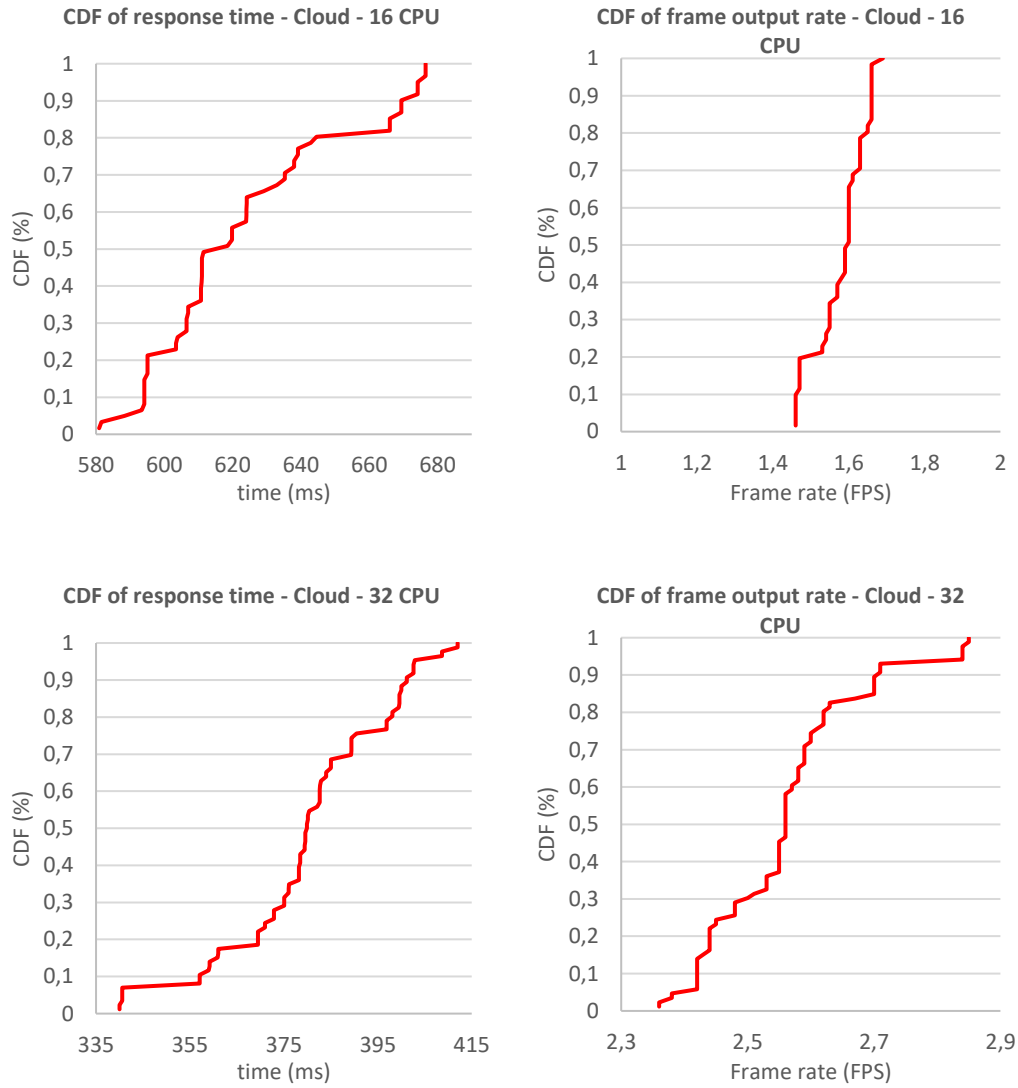


Figure 53 - CDF for response time and frame rate for FaceSwap - Clouds

6.2.3. RESULTS

Figure 54 presents the comparison between the Cloudlet and Cloud for the offloading of part of Fluid application on the servers. The quality of the graphics is much higher in Cloudlet scenario, for Cloudlet response time median values are around 38-43 ms and for Cloud the median values are around 205-618 ms. For the same VM instance types launched, 2 and 4 CPU, latency values almost lower 5 times than the values measured for Cloud AWS EC2 Oregon. Using the cloudlet server face recognition is so much faster.

We notice a bandwidth reduction and maybe some losses on AWS for experiments regarding 16 and 32 CPU cases. We tried two times this experiment with an interval of 1 hour and the results maintained similarly.

The other key feature analyzed is the output frame rate that presents the median values of 19-22 FPS for Cloudlet and 2-5 for Cloud. Using Cloudlet ensures a rate of at least 4 times higher than the Cloud, so it will satisfy user experience with the Cloudlet scenario.

This 2 features analyzed to ensure that the use of Cloudlet can have a great impact and reduce brutally the latency and augment substantially the user quality of experience.

To use this application and collect an application capable of performing face swapping, the recommended FaceSwap server should have 4 cores and 8GM RAM [96]. The values presented by the Cloudlet successfully accomplish the application requirements. We also noticed that the values collected by J. Wang are similar compared to the results collected in our experiments.

Compared with some other works using the application Face [98] [99] [100], the results obtained in our experiments presents lower response time both for Cloudlet and AWS West solutions. In all experiments, Cloudlets present better response time values than the cloud solutions.

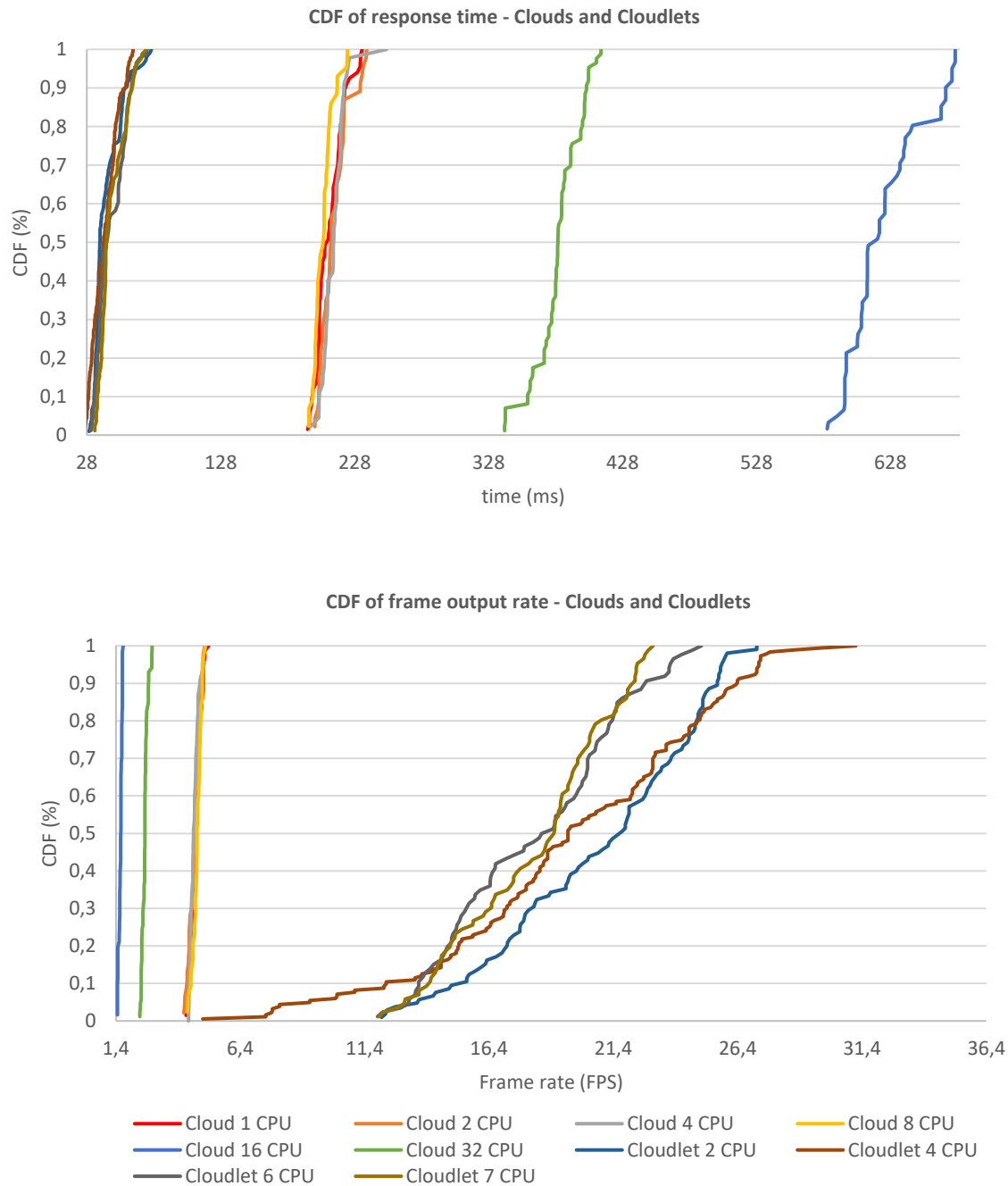


Figure 54 - Cloud and Cloudlet test result comparison for FaceSwap

It was analyzed the performance on launching VM instance with a higher configuration of CPU and RAM intensive computation tasks on the server side. We noticed also that for FaceSwap application launched on VM instances with 6 and 7 CPU, the CPU of the original workstation was surcharging and even shut down, as shown in Figure 55. The "Stealth time" (st) stands for the amount of CPU that has been allocated by the hypervisor to the virtual machine that is not being utilized by the virtual machine. As this value should be 0 outside

a virtualized environment, we can assume that the issue is between the VM instantiated and the hypervisor. Libvirt is a toolkit used for communication with the hypervisor qemu-KVM. It seems that the VM Instantiated is reaching almost 100% of each vCPU during the phase of face recognition as show in Figure 55.

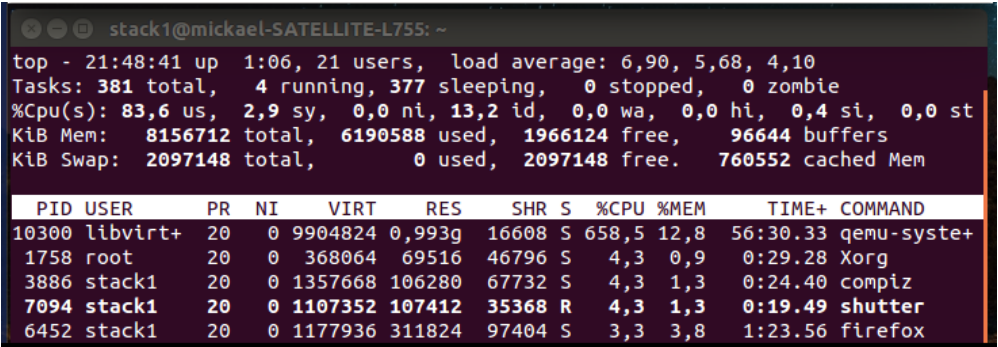


Figure 55 - CPU usage while FaceSwap running

7. CONCLUSION

This thesis presents an emerging technology that enables the reduction of the end-to-end latencies and the increase of the user quality when using applications with mobile devices. MEC is not replacing but complimenting the cloud computing model. The delay sensitive part of application can be executed on MEC server, whereas delay tolerant compute intensive part of application can be executed on the remote cloud server. MEC aims to enable the billions of connected mobile devices to execute the real time compute intensive applications directly at the network edge.

We prove that a MEC server can improve user interaction and quality of experience by offloading the processing and computation intensive on edge servers. The values measured in both use case scenario are convincing as the latency is always under 100 ms and always better than the ones presented on remote servers (AWS EC2-Oregon). Compared to the remote servers (AWS EC2-Oregon), tests realized showed that a MEC solution achieves latency of about 25% of the MCCs latency and about 2 to 5 higher frame rate than in MCC. Its proximity to the user allows a lower response time and delivers a better user experience.

The results of the VM synthesis show that resuming an instance from an overlay is a fast operation nearby cloudlet. New emerging applications runs on WiFi and LTE networks will improve greatly that results and prove even more the results.

We analyzed also the performance in the response time due to the variation of the server processing capacity. This was achieved by considering several cores at the Cloudlet. In the two uses cases (Fluid and Faceswap), the benefits were not really significant because those applications were implemented using a single thread aproach.

While 5G technology is already on the way, MEC has a great role to play on the mobile ecosystem with the increase of a new application like face recognition, real-time online games, IoT, which are interactive and compute-intensive.

For future work, we propose an analysis on the pertinent interactions between MEC with RAN through the selection and the traffic control in the User Plane. An interesting analysis can be performed through the evaluation of the services performance and continuity in a UE mobility environment.

References

- [1] M. Armbrust, R. G. A. Fox, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," Feb. 2012. [Online]. Available: <https://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf> 32
- [2] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: State-of-the-art and research challenges," *Journal Internet Services appl.*, vol. 1, no. 1, pp. 7–18, 2010.
- [3] "More than 50 billion connected devices," Ericsson, February 2011. [Online]. Available: <http://www.ericsson.com/res/docs/whitepapers/wp-50-billions.pdf>
- [4] M. Skarpness, "Keynote: Beyond the Cloud: Edge Computing", Open Source Summit.
- [5] ETSI, "Mobile-edge computing introductory technical white paper," White Paper, Mobile-edge Computing Industry Initiative. [Online]. Available: https://portal.etsi.org/portals/0/tbpages/mec/docs/mobile-edge_computing_introductory_technical_white_paper_v1%2018-09-14.pdf
- [6] Juniper, "Smart wireless devices and the Internet of me," White paper, Mar. 2015. [Online]. Available: <http://itersnews.com/wp-content/uploads/experts/2015/03/96079Smart-Wireless-Devices-and-the-Internet-of-Me.pdf>
- [7] CISCO, "The Internet of Things how the next evolution of the Internet is changing everything," White paper, Apr. 2011. [Online]. Available: http://www.cisco.com/c/dam/en us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf
- [8] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. PP, no. 99, pp. 1–1, 2016.
- [9] GSMA INTELLIGENCE, "ANALYSIS - Understanding 5G: Perspectives on future technological advancements in mobile", White Paper, Dec. 2014. [Online]. Available: <https://www.gsmainelligence.com/research/?file=141208-5g.pdf&download>
- [10] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta and P. Popovski, "Five disruptive technology directions for 5G," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 74-80, Feb. 2014.

- [11] H. T. Dinh, C. Lee, D. Niyato, P. Wang, A Survey of Mobile Cloud Computing: Architecture, Applications, and Approaches, *Wireless Communications and Mobile Computing* 13 (18) (2013) 1587–1611.doi:10.1002/wcm.1203.
- [12] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, “The case for VM-based cloudlets in mobile computing,” *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, 2009.
- [13] G. Intelligence, “Understanding 5G: Perspectives on future technological advancements in mobile,” London, UK, 2014.
- [14] A. Somov and R. Giaffreda, “Powering IoT devices: Technologies and opportunities,” Available: <http://iot.ieee.org/newsletter/november-2015/powering-iot-devices-technologies-and-opportunities.html>.
- [15] R. Kemp, N. Palmer, T. Kielmann, F. Seinstra, N. Drost, J. Maassen, and H. Bal, “EyeDentify: Multimedia cyber foraging from a smartphone,” in *Proc. IEEE Int. Symp. Multimedia*, San Diego, CA, USA, Dec. 2009, pp. 392–399.
- [16] B. Shi, J. Yang, Z. Huang, and P. Hui, “Offloading guidelines for augmented reality applications on wearable devices,” in *Proc. ACM Int. Symp. Multimedia*, Brisbane, Australia, Oct. 2015, pp. 1271–1274.
- [17] W. N. Schilit, “A system architecture for context-aware mobile computing,” Ph.D. dissertation, Columbia University, 1995.
- [18] S. Nunna, A. Kousaridas, M. Ibrahim, M. Dillinger, C. Thuemmler, H. Feussner, and A. Schneider, “Enabling real-time context-aware collaboration through 5G and mobile edge computing,” in *Proc. IEEE Int. Conf. Inf. Techn. New Generations (ITNG)*, Las Vegas, NV, Apr. 2015, pp. 601–605.
- [19] X. Luo, “From augmented reality to augmented computing: A look at cloud-mobile convergence,” in *Proc. IEEE Int. Symp. Ubiquitous Virtual Reality*, Gwangju, South Korea, Jul. 2009, pp. 29–32.
- [20] A. Thiagarajan, L. Ravindranath, H. Balakrishnan, S. Madden, and L. Girod, “Accurate, low-energy trajectory mapping for mobile devices,” in *Proc. USENIX Symp. Networked Systems Design and Implementation (NSDI)*, Boston, MA, Mar. 2011, pp. 1–14.

- [21] H. Suo, Z. Liu, J. Wan, and K. Zhou, "Security and privacy in mobile cloud computing," in Proc. IEEE Int. Wireless Commun. Mobile Comput. Conf. (IWCMC), Cagliari, Italy, Jul. 2013, pp. 655–659.
- [22] ETSI, "Mobile-edge computing (MEC): Service scenarios." [Online]. Available: http://www.etsi.org/deliver/etsi_gs/MEC-IEG/001_099/004/01.01.01_60/gs_mec-ieg004v010101p.pdf
- [23] Y. Mao, C. You, J. Zhang, K. Huang and K. Letaiefet, "A survey on mobile edge computing: The communication perspective". IEEE Communications Surveys & Tutorials, 2017. 19(4): p. 2322-2358.
- [24] Juniper, "White paper: Mobile edge computing use cases & deployment options." [Online]. Available: <https://www.juniper.net/assets/us/en/local/pdf/whitepapers/2000642-en.pdf>
- [25] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A overview of cloud computing," Commun. ACM, vol. 53, no. 4, pp. 52–58, Apr. 2010.
- [26] M. Othman, S. A. Madani, S. U. Khan et al., "A survey of mobile cloud computing application models," IEEE Commun. Surveys Tuts., vol. 16, no. 1, pp. 393–413, 1st Quater 2014.
- [27] M. F. Bari, R. Boutaba, R. Esteves, L. Z. Granville, M. Podlesny, M. G. Rabbani, Q. Zhang, and M. F. Zhani, "Data center network virtualization: A survey," IEEE Commun. Surveys Tuts., vol. 15, no. 2, pp. 909–928, 2nd Quarter 2013.
- [28] A. Ghiasi and R. Baca, "Overview of largest data centers," IEEE 802.3bs Task Force Interim Meeting, May 2014. [Online]. Available: http://www.ieee802.org/3/bs/public/14_05/ghiasi_3bs_01b_0514.pdf
- [29] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher and V. Young, "Mobile Edge Computing, A Key Technology Towards 5G, " ETSI White Paper, 2015.
- [30] S. Clinch, J. Harkes, A. Friday, N. Davies and M. Satyanarayanan, "How close is close enough? Understanding the role of cloudlets in supporting display appropriation by mobile users," in Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom), Lugano, Switzerland, Mar. 2012, pp. 122–127.

- [31] Tuyen X Tran, Abolfazl Hajisami, Parul Pandey, and Dario Pompili. Collaborative mobile edge computing in 5g networks: New paradigms, scenarios, and challenges. *IEEE Communications Magazine*, 55(4):54–61, 2017.
- [32] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, “A survey on mobile edge networks: Convergence of computing, caching and communications,” *IEEE Access*, to appear.
- [33] J. Zhang, W. Xie, F. Yang, and Q. Bi, “Mobile edge computing and field trial results for 5G low latency scenario,” *China Commun.*, vol. 13, no. 2 (Supplement), pp. 174–182, 2016.
- [34] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, “Maui: Making smartphones last longer with code offload,” in *Proc. ACM Int. Conf. Mobile Syst. Appl. Serv. (MobiSys)*, San Francisco, California, USA, Jun. 2010, pp. 49–62.
- [35] 5GPPP, “5g automotive vision,” White Paper. [Online]. Available: <https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP-White-Paper-on-Automotive-Vertical-Sectors.pdf>
- [36] O. Khalid, M. Khan, S. Khan, and A. Zomaya, “OmniSuggest: A ubiquitous cloud based context aware recommendation system for mobile social networks,” *IEEE Trans. Serv. Comput.*, vol. 7, no. 3, pp. 401–414, Dec. 2014.
- [37] K. Goel and M. Goel, “Cloud computing based e-commerce model,” in *Proc. IEEE Int. Conf. Recent Trends in Electron., Info. & Commun. Techn. (RTEICT)*, Bangalore, India, May 2016, pp. 27–30.
- [38] G. Riah, “E-learning systems based on cloud computing: A review,” *ELSEVIER Proc. Comput. Sci.*, vol. 62, pp. 352–359, Sep. 2015.
- [39] A. Abbas and S. U. Khan, “A review on the state-of-the-art privacy preserving approaches in the e-health clouds,” *IEEE J. Biomed. Health Inform.*, vol. 18, no. 4, pp. 1431–1441, Apr. 2014.
- [40] M. T. Beck, M. Werner, S. Feld and S. Schimper, “Mobile edge computing: A taxonomy,” in *Proc. of the Sixth International Conference on Advances in Future Internet*, Citeseer, 2014, pp. 48–54

- [41] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless communications and mobile computing*, vol. 13, no. 18, pp. 1587-1611, Dec. 2013.
- [42] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog Computing and its Role in the Internet of Things, in: *Proceedings of the 1st Edition of the MCC Workshop on Mobile Cloud Computing*, 2012, pp. 13–16. doi:10.1145/2342509.2342513.
- [43] Shenoy K, Bhokare P, Pai U (2015) Fog computing future of computing. *Int J Sci Res* 4(6):55–56
- [44] Suryawanshi R, Mandlik G. Focusing on mobile users at edge and internet of things using fog computing (2015)
- [45] Aazam M, Huh E-N (2014) Fog computing and smart gateway based communication for cloud of things. In: *Future internet of things and cloud (FiCloud)*, 2014 international conference on. IEEE, pp 464–470
- [46] F. Bonomi, R. Milito, P. Natarajan, J. Zhu, Fog Computing: A Platform for Internet of Things and Analytics, in: N. Bessis, C. Dobre (Eds.), *Big Data and Internet of Things: A Roadmap for Smart Environments*, Vol. 546 of *Studies in Computational Intelligence*, Springer International Publishing, 2014, pp. 169–186. doi:10.1007/978-3-319-05029-4_7.
- [47] L. M. Vaquero, L. Roderio-Merino, Finding Your Way in the Fog: Towards a Comprehensive Definition of Fog Computing, *SIGCOMM Comput. Commun. Rev.* 44 (5) (2014) 27–32. doi:10.1145/2677046.2677052.
- [48] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, M. Satyanarayanan, Towards Wearable Cognitive Assistance, in: *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2014, pp. 68–81. doi:10.1145/2594368.2594383.
- [49] S. Jingtao, L. Fuhong, Z. Xianwei, L. Xing, Steiner Tree based Optimal Resource Caching Scheme in Fog Computing, *China Communications* 12 (8) (2015) 161–168. doi:10.1109/CC.2015.7224698.
- [50] O. T. T. Kim, N. D. Tri, V. D. Nguyen, N. Tran, C. S. Hong, A Shared Parking Model in Vehicular Network using Fog and Cloud Environment, in: *Proceedings of the 17th*

- Asia-Pacific Network Operations and Management Symposium (APNOMS), 2015, pp. 321–326. doi:10.1109/APNOMS.2015.7275447.
- [51] J. Zao, T. T. Gan, C. K. You, S. Rodriguez Mendez, C. E. Chung, Y. T. Wang, T. Mullen, T. P. Jung, Augmented Brain Computer Interaction Based on Fog Computing and Linked Data, in: Proceedings of the International Conference on Intelligent Environments (IE), 2014, pp. 374–377. doi:10.1109/IE.2014.54.
 - [52] M. T. Beck, M. Maier, “Mobile edge computing: challenges for future virtual network embedding algorithms”, in: Proc. The Eighth International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 2014)
 - [53] Elijah: Cloudlet-based Edge Computing. <http://elijah.cs.cmu.edu/> Accessed 10 June 2018
 - [54] G. I. Klas, “Fog computing and mobile edge cloud gain momentum open fog consortium etsi mec and cloudlets”, 2015, [online] Available: <http://yucianga.info/?p=938>.
 - [55] Khan KA, Wang Q, Luo C, Wang X, Grecos C (2014) Comparative study of internet cloud and cloudlet over wireless mesh networks for real-time applications. In: SPIE Photonics Europe. International Society for Optics and Photonics, pp 91390–91390
 - [56] OpenEdgeComputing. <http://openedgecomputing.org/> . Accessed 19 July 2017
 - [57] Qing W, Zheng H, Ming W, Haifeng L (2013) Cactse: cloudlet aided cooperative terminals service environment for mobile proximity content delivery. China Commun 10(6):47–59
 - [58] Soyata T, Muraleedharan R, Funai C, Kwon M, Heinzelman W (2012) Cloudvision: real-time face recognition using a mobile-cloudlet-cloud acceleration architecture. In: Computers and communications (ISCC), 2012 IEEE symposium on. IEEE, pp 000059–000066
 - [59] Achanta VS, Sureshbabu NT, Thomas V, Sahitya ML, Rao S (2012) Cloudlet-based multi-lingual dictionaries. In: Services in Emerging Markets (ICSEM), 2012 third international conference on. IEEE, pp 30–36
 - [60] Koukoumidis E, Lymberopoulos D, Strauss K, Liu J, Burger D (2011) Pocket cloudlets. ACM SIGARCH computer architecture news. ASPLOS XVI Proceedings

of the sixteenth international conference on Architectural support for programming languages and operating systems 39(1):171–184. Available: <https://dl.acm.org/citation.cfm?id=1950387>

- [61] Verbelen T, Simoens P, De Turck F, Dhoedt B (2012) Cloudlets: bringing the cloud to the mobile user. In: Proceedings of the third ACM workshop on mobile cloud computing and services. ACM, pp 29–36
- [62] Abolfazli S, Sanaei Z, Ahmed E, Gani A, Buyya R., “Cloud-based augmentation for mobile devices: motivation, taxonomies, and open challenges”. CoRR, vol. abs/1306.4956, 2013
- [63] ETSI, “Mobile-edge computing (MEC): Terminology.” [Online]. Available: http://www.etsi.org/deliver/etsi_gs/MEC/001_099/001/01.01.01_60/gs_MEC001v010101p.pdf 2
- [64] ETSI, Mobile Edge Computing (MEC): Technical Requirements.” [Online]. Available: http://www.etsi.org/deliver/etsi_gs/MEC/001_099/002/01.01.01_60/gs_MEC002v010101p.pdf
- [65] ETSI, “Mobile-edge computing (MEC): Framework and reference architecture.” [Online]. Available: http://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/01.01.01_60/gs_MEC003v010101p.pdf
- [66] ETSI, “Mobile-edge computing (MEC): Service scenarios.” [Online]. Available: http://www.etsi.org/deliver/etsi_gs/MEC/001_099/002/01.01.01_60/gs_MEC002v010101p.pdf
- [67] ETSI, “Mobile Edge Computing (MEC) Proof of Concept Framework.” [Online]. Available: http://www.etsi.org/deliver/etsi_gs/MEC-IEG/001_099/005/01.01.01_60/gs_MEC-IEG005v010101p.pdf
- [68] ETSI, “Mobile Edge Computing: Market Acceleration MEC Metrics Best Practice and Guidelines.” [Online]. Available: http://www.etsi.org/deliver/etsi_gs/MEC-IEG/001_099/006/01.01.01_60/gs_MEC-IEG006v010101p.pdf
- [69] N. Sprecher, J. Friis, R. Dolby, and J. Reister, “Edge computing prepares for a multi-access future”, Sep.2016. [Online]. Available:

<http://www.telecomtv.com/articles/mec/edge-computing-prepares-for-a-multi-access-future-13986/>

- [70] 3GPP, “Technical specification group services and system aspects; system architecture for the 5g systems; stage 2 (release 15),” 3GPP TS 23.501 V0.4.0, Apr. 2017.[Online].Available:<https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144>
- [71] [71] ETSI, “Mobile Edge Computing (MEC); General principles for Mobile Edge Service APIs”. [Online]. Available:http://www.etsi.org/deliver/etsi_gs/MEC/001_099/009/01.01.01_60/gs_MEC009v010101p.pdf
- [72] ETSI, “Mobile Edge Computing (MEC); Mobile Edge Management; Part 2: Application lifecycle, rules and requirements management” [Online]. Available: http://www.etsi.org/deliver/etsi_gs/MEC/001_099/01002/01.01.01_60/gs_MEC01002v010101p.pdf
- [73] ETSI, “Mobile Edge Computing (MEC); Mobile Edge Platform Application Enablement” [Online]. Available: http://www.etsi.org/deliver/etsi_gs/MEC/001_099/011/01.01.01_60/gs_MEC011v010101p.pdf
- [74] TSI, “Mobile Edge Computing (MEC); Radio Network Information API” [Online].Available:http://www.etsi.org/deliver/etsi_gs/MEC/001_099/012/01.01.01_60/gs_MEC012v010101p.pdf
- [75] ETSI, “Mobile Edge Computing (MEC); Location API” [Online]. Available: http://www.etsi.org/deliver/etsi_gs/MEC/001_099/013/01.01.01_60/gs_MEC013v010101p.pdf
- [76] ETSI, “Mobile Edge Computing (MEC); UE application interface” [Online]. Available:http://www.etsi.org/deliver/etsi_gs/MEC/001_099/016/01.01.01_60/gs_MEC016v010101p.pdf
- [77] ETSI, “Mobile Edge Computing (MEC); Mobile Edge Management; Part 1: System, host and platform management” [Online]. Available:

http://www.etsi.org/deliver/etsi_gs/MEC/001_099/01001/01.01.01_60/gs_MEC01001v010101p.pdf

- [78] ETSI, “Mobile Edge Computing(MEC); Bandwidth Management API” [Online]. Available:http://www.etsi.org/deliver/etsi_gs/MEC/001_099/015/01.01.01_60/gs_MEC015v010101p.pdf
- [79] ETSI, “Mobile Edge Computing (MEC); End to End Mobility Aspects” [Online]. Available:http://www.etsi.org/deliver/etsi_gr/MEC/001_099/018/01.01.01_60/gr_MEC018v010101p.pdf
- [80] ETSI, “Mobile Edge Computing (MEC); UE Identity API” [Online]. Available: http://www.etsi.org/deliver/etsi_gs/MEC/001_099/014/01.01.01_60/gs_MEC014v010101p.pdf
- [81] ETSI, “Mobile Edge Computing (MEC); Deployment of Mobile Edge Computing in an NFV environment” [Online]. Available: http://www.etsi.org/deliver/etsi_gr/MEC/001_099/017/01.01.01_60/gr_MEC017v010101p.pdf
- [82] M. Satyanarayanan, G. Lewis, E. Morris, S. Simanta, J. Boleng and K. Ha, “The role of cloudlets in hostile environments,” *Pervasive Computing*, vol. 4, pp. 40-49, 2013.
- [83] S. Simanta, G. Lewis, E. Morris, K. Ha and M. Satyanarayanan, “Cloud computing at the tactical edge,” 2012.
- [84] S. Simanta, K. Ha, G. Lewis, E. Morris and M. Satyanarayanan, “A reference architecture for mobile code offload in hostile environment,” *Mobile Computing, Applications, and Services*, pp. 274-293, 2013.
- [85] Open Edge Computing elijah-OpenStack, Online available: <https://libraries.io/github/OpenEdgeComputing/elijah-openstack>
- [86] B. Solenthaler, R. Pajarola. Predictive-corrective incompressible SPH. *ACM Trans. Graph.*, 28(3):40:1–40:6, July 2009. ISSN 0730-0301. doi: 10.1145/1531326.1531346. URL <http://doi.acm.org/10.1145/1531326.1531346> .
- [87] “Galaxy S8”, available online, <https://www.samsung.com/global/galaxy/galaxy-s8/specs/>

- [88] “Toshiba Satellite L755-1DR”, available online, <http://www.toshiba.pt/discontinued-products/satellite-l755-1dr/>
- [89] B. Solenthaler, R. Pajarola. Predictive-corrective incompressible SPH. ACM Trans. Graph., 28(3):40:1–40:6, July 2009. ISSN 0730-0301. doi: 10.1145/1531326.1531346, available online: <http://doi.acm.org/10.1145/1531326.1531346>.
- [90] FaceSwap Android Client, available online: <https://play.google.com/store/apps/details?id=edu.cmu.cs.faceswap>
- [91] K. Ha, C. Z, W. Hu, W. Richter, P. Pillai and M. Satyanarayanan, “Towards wearable cognitive assistance,” in MobiSys '14 Proceedings of the 12th annual international conference on Mobile systems, applications, and services, New York, NY, USA, 2014.
- [92] OpenFace. Available online: <https://cmusatyalab.github.io/openface/demo-3-classifier/>
- [93] Matthew Turk and Alex Pentland. Eigenfaces for recognition. Journal of Cognitive Neuroscience, 3(1):71–86, 1991.
- [94] OpenCV. OpenCV Wiki. <http://opencv.willowgarage.com/wiki/>.
- [95] K. Ha, “System Infrastructure for Mobile-Cloud Convergence”. PhD thesis, Carnegie Mellon University Pittsburgh, PA, 2016.
- [96] FaceSwap Github, available Online: <https://cmusatyalab.github.io/faceswap/dev-guide/>
- [97] OPNFV, available Online: <https://www.opnfv.org/>
- [98] Z. Chen, W. Hu, J. Wang, S. Zhao, B. Amos2, G. Wu, K. Ha, K. Elgazzar, P. Pillai, R. Klatzky, D. Siewiorek, M. Satyanarayanan, An empirical study of latency in an emerging class of edge computing applications for wearable cognitive assistance, SEC 2017: 14:1-14:14
- [99] K. Ha, P. Pillai, G. Lewis, S. Simanta, S. Clinch, N. Davies, M. Satyanarayanan, The Impact of Mobile Multimedia Applications on Data Center Consolidation. IC2E 2013: 166-176

- [100] W. Hu, Y. Gao, K. Ha, J. Wang, B. Amos, Z. Chen, P. Pillai, M. Satyanarayanan, Quantifying the Impact of Edge Computing on Mobile Applications. APSys 2016: 5:1-5:8
- [101] M. Satyanarayana, et al., "Cloudlets: at the Leading Edge of Mobile-Cloud Convergence", MobiCASE 2014.
- [102] K. Dolui, S. Datta, "Comparison of edge computing implementations: Fog computing cloudlet and mobile edge computing", GIoTS, 2017.
- [103] Y. Ai, M. Peng, K. Zhang, Edge cloud computing technologies for internet ofthings: A primer, Digital Communications and Networks (2017), doi: 10.1016/j.dcan.2017.07.001
- [104] D. Sabella, A. Vaillant, P. Kuure, Pekka U. Rauschenbach, F. Giust, Fabio. Mobile-Edge Computing Architecture: The role of MEC in the Internet of Things., 2016, IEEE Consumer Electronics Magazine. 5. 84-91. 10.1109/MCE.2016.2590118.
- [105] T. Tran, A. Hajisami, P. Pandey, D. Pompili, Dario, Collaborative Mobile Edge Computing in 5G Networks: New Paradigms, Scenarios, and Challenges, 2017, IEEE Communications Magazine. 55. 10.1109/MCOM.2017.1600863.
- [106] J. Okwuibe, M. Liyanage, I. Ahmad, M. Ylianttila, Mika. Cloud and MEC security, 2018, 10.1002/9781119293071.ch16.
- [107] ETSI, "MEC Deployments in 4Gand Evolution Towards 5G", ETSI White Paper No. 24, [Online]. Available: https://www.etsi.org/images/files/etsiwhitepapers/etsi_wp24_mec_deployment_in_4g_5g_final.pdf
- [108] ETSI, "MEC in 5G networks", ETSI White Paper No. 28, [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp28_mec_in_5G_FINAL.pdf

History

- 5 de Setembro de 2018, Versão 1.0, <mailto:1000167@isep.ipp.pt>
- 30 de Outubro de 2018, Versão 1.1, <mailto:1000167@isep.ipp.pt>
- 11 de Março de 2019, Versão 1.2, [mailto: 100167@isep.ipp.pt](mailto:100167@isep.ipp.pt)
- 21 de Junho de 2019, Versão 1.7, [mailto: 100167@isep.ipp.pt](mailto:100167@isep.ipp.pt)
- 3 de Julho de 2019, Versão 1.10, [mailto: 100167@isep.ipp.pt](mailto:100167@isep.ipp.pt)